A

Using Generalized Annotated Programs to Solve Social Network Diffusion Optimization Problems

PAULO SHAKARIAN, United States Military Academy West Point
MATTHIAS BROECHELER, University of Maryland
V.S. SUBRAHMANIAN, University of Maryland
CRISTIAN MOLINARO, Università della Calabria

There has been extensive work in many different fields on how phenomena of interest (e.g. diseases, innovation, product adoption) "diffuse" through a social network. As social networks increasingly become a fabric of society, there is a need to make "optimal" decisions with respect to an observed model of diffusion. For example, in epidemiology, officials want to find a set of k individuals in a social network which, if treated, would minimize spread of a disease. In marketing, campaign managers try to identify a set of k customers that, if given a free sample, would generate maximal "buzz" about the product. In this paper, we first show that the well-known Generalized Annotated Program (GAP) paradigm can be used to express many existing diffusion models. We then define a class of problems called *Social Network Diffusion Optimization Problems* (SNDOPs). SNDOPs have four parts: (i) a diffusion model expressed as a GAP, (ii) an objective function we want to optimize with respect to a given diffusion model, (iii) an integer $k > 0$ describing resources (e.g. medication) that can be placed at nodes, (iv) a logical condition VC that governs which nodes can have a resource (e.g. only children above the age of 5 can be treated with a given medication). We study the computational complexity of SNDOPs and show both NP-completeness results as well as results on complexity of approximation. We then develop an exact and a heuristic algorithm to solve a large class of SNDOP problems and show that our GREEDY-SNDOP algorithm achieves the best possible approximation ratio that a polynomial algorithm can achieve (unless $\mathbf{P} = \mathbf{NP}$). We conclude with a prototype experimental implementation to solve SNDOPs that looks at a real-world Wikipedia data set consisting of over 103,000 edges.

Categories and Subject Descriptors: I.2.4 [**Knowledge Representation Formalisms and Methods**]: Representations (procedural and rule-based); I.2.3 [**Deduction and Theorem Proving**]: Logic programming

Additional Key Words and Phrases: Social Network, Generalized Annotated Programs, Approximation Algorithms

ACM Reference Format:
ACM Trans. Comput. Logic V, N, Article A (January YYYY), 40 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

There is a rapid proliferation of different types of *graph data* in the world today. These include social network data (FaceBook, Flickr, YouTube, etc.), cell phone network data [N. Eagle and Lazer 2008] collected by virtually all cell phone vendors, email

Author's addresses: P. Shakarian; Network Science Center and Department of Electrical Engineering and Computer Science, United States Military Academy West Point; M. Broecheler and V.S. Subrahmanian, Department of Computer Science, University of Maryland; Cristian Molinaro, DEIS Department, Università della Calabria.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2013		**00-00-2013 to 00-00-2013**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Using Generalized Annotated Programs to Solve Social Network Diffusion Optimization Problems	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **United States Military Academy,West Point,NY,10996**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
ACM Transactions on Computational Logic, Vol. V, No. N, Article A, Publication date: January YYYY

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**53**	

network data (such as those derived from the Enron corpus[1]), as well as information on disease networks [Coelho et al. 2008; Anderson and May 1979]. In addition, the World Wide Consortium's RDF standard is also a graph-based standard for encoding semantic information contained in web pages. There has been years of work on analyzing how various properties of nodes in such networks "diffuse" through the network - different techniques have been invented in different academic disciplines including economics [Jackson and Yariv 2005; Schelling 1978], infectious diseases [Coelho et al. 2008], sociology [Granovetter 1978] and computer science [Kempe et al. 2003].

Past work on diffusion has several limitations. (i) First, they largely assume that a social network is nothing but a set of vertices and edges [Watts 1999; Cowan and Jonard 2004; Rychtář and Stadler 2008]. In contrast, in this paper we adopt a richer model where edges and vertices can both be labeled with properties. For instance, a political campaigner hoping to spread a positive message about a campaign needs to use demographics (e.g. sex, age group, educational level, group affiliations, etc.) for targeting a political message — a "one size fits all" message will not work. In general, social network researchers would say that they have several sociomatrices that can be used for such applications. (ii) Second, past work on diffusion has no notion of "strength" associated with edges. It may well be the case, in many applications, that the degree of contact between two vertices (e.g. number of minutes person A spends on the cell phone with person B) is a proxy for the strength of the relationship between A and B, which in turn may have an impact of whether A can influence B or not. (iii) Third, these past frameworks [Jackson and Yariv 2005; Schelling 1978; Coelho et al. 2008; Granovetter 1978] usually reason about a single diffusion model, rather than develop a framework for reasoning about a whole class of diffusion models.

Past diffusion models developed in a variety of fields ranging from business [Jackson and Yariv 2005], economics [Schelling 1978], social science [Granovetter 1978], epidemiology [Coelho et al. 2008; Hethcote 1976; Anderson and May 1979], mobile phone usage [Aral et al. 2009] show that diffusion models vary dramatically from application to application. Three broad categories of diffusion models exist.

(1) *Cascade models* [Coelho et al. 2008; Hethcote 1976; Anderson and May 1979] are widespread in epidemiology and assume that diffusions are largely based on connectivity between nodes and are largely probabilistic.
(2) *Tipping models* do not use probabilities, but use various quantitative calculations to determine when a vertex adopts (or is infected with) a diffusive property. They are omnipresent in the social sciences and business [Centola 2010; Jackson and Yariv 2005; Granovetter 1978]. Nobel-laureate Tom Schelling makes a similar point that diffusions in many social science applications have a tipping point when vertices become influenced by the number of neighbors and the strength of commitment the neighbors may have to a certain position. No probabilities are present in such models.
(3) *Homophilic models* are ones where similarity between users, rather than networks effects, dominate diffusion. Similarity is usually calculated using some quantitative model, often related to distance between vectors representing (values of) properties of nodes. For example, [Aral et al. 2009] tracks adoption of mobile applications in a study of over 27M users and shows that homophily - similarity between users - is the most compelling diffusion model. There are no probabilities here, just similarity measures. Another world famous diffusion model focused on marketing [Watts and Peretti 2007] also is based on homophily and similarity of nodes' intrinsic properties rather than a probability.

[1] http://www.cs.cmu.edu/~enron/

Moreover, many models use a mix of the above forms. For instance, [Cha et al. 2009] argues that the way photos are marked as "favorites" on Flickr is based on a mix of cascading and homophilic behavior and to study the former, one must also account for the latter. A similar combination of cascading and tipping is observed in [Zhang 2011]. Another strong indication of hybrid models in real social networks is the noteworthy experimental study of [Centola 2011] which illustrates how a tipping model combined with homphilic effects promote diffusion of health behaviors in an online network. Thus, *any general framework for expressing diffusions must have the capability to express all three types of diffusion models*, not just one or the other. In general, a language to express diffusion models must be capable of expressing a wide variety of *quantitative* methods encapsulated in the above.

In this paper, we first show that a class of the well-known Generalized Annotated Program (GAP) paradigm [Kifer and Subrahmanian 1992; Kifer and Lozinskii 1992; Thirunarayan and Kifer 1993] and their variants [Vennekens et al. 2004; Krajci et al. 2004; Lu 1996; Lu et al. 1993; Damasio et al. 1999] including *Linear GAPs* (introduced here) form a convenient method to express many diffusion models. Though there is no claim that they can express all possible useful diffusion models, they do express all diffusion models (over 30) we have studied in the literature on a wide variety of topics. Moreover, [Broecheler et al. 2010] provides an algorithm to automatically learn such diffusion models from historical data, so users do not need to write their diffusion models by themselves. This provides greater confidence that these diffusion models are "correct." Many other papers also focus on learning diffusion models automatically for different types of applications — [Leskovec et al. 2007a] develop a probabilistic learning algorithm, while [Backstrom et al. 2006] develop a method that takes both the properties of vertices and the strength of relationships between vertices to learn such a diffusion model automatically. We expect that in most real-world applications going forward, diffusion models will be automatically learned rather than being programmed by logic programmers.

Next, unlike most existing work in social networks which focus on learning diffusion models, we focus on reasoning with diffusion models (expressed via GAPs) *after the diffusion models have been learned*. In particular, we consider the problem of optimal decision making in social networks which have associated diffusion models expressible as Linear GAPs, though many of the results in the paper apply to arbitrary GAPs as well. Here are two examples.

— **(Q1) Cell phone plans.** A cell phone company is promoting a new cell phone plan - as a promotion, it is giving away k free plans to existing customers.[2] Which set of k people should they pick so as to maximize the number of plan adoptees predicted by a cell phone plan adoption diffusion model they have learned from their past promotions?
— **(Q2) Medication distribution plan.** A government combating a disease spread by physical contact has limited stocks of free medication to give away. Based on a diffusion model of how the disease spreads (e.g. kids might be more susceptible than adults, those previously inoculated against the disease are safe, etc.), they want to find a set of k people who (jointly) maximally spread the disease when infected (so that they can provide immediate treatment to these k people in an attempt to

[2]Our framework allows us to add additional constraints — for instance, that plans can only be given to customers satisfying certain conditions, e.g.customers deemed to be "good" according to various business criteria.

halt the disease's spread).[3] Notice that this query corresponds to only one of many different policies that can be considered to deal with the disease spread scenario, that is, we consider the case where a diffusion model expressing how an infected person can infect other people is available and formulate a query that looks at the maximum spread when k people are infected. Other queries, possibly leading to different answers about who should be treated with medications, are possible.

Both these problems are instances of a class of queries that we call *Social Network Diffusion Optimization Problem* (SNDOP) queries. They differ from queries studied in the past in quantitative (both probabilistic and annotated) logic programming in two fundamental ways: (i) They are specialized to operate on graph data where the graph's vertices and edges are labeled with properties and where the edges can have associated weights, (ii) They find *sets of vertices* that optimize complex objective functions that can be specified by the user. Neither of these has been studied before by any kind of quantitative logic programming framework, though work on optimizing objective functions in the context of different types of semantics (minimal model and stable model semantics) has been studied before [Leone et al. 2004]. And of course, constraint logic programming [Apt 2003] has also extensively studied optimization issues as well in logic programming - however, here, optimization and constraint solving is embedded in the constraint logic program, whereas in our case, they are part of the *query* over an annotated logic program. Moreover, most measures of importance in social networks are *centrality measures* that study the influence of single vertices - [Borgatti and Everett 2006] provides an excellent overview of centrality measures. In contrast, a set of k nodes each with low individual centrality may often wield greater influence on a network than the set consisting of the k nodes with highest individual centrality - intuitively, this is due to the fact that the k nodes with highest individual centrality may overlap greatly in the nodes they influence, leading to an aggregate number of influenced nodes that is lower than the one in the first case.

This paper is organized as follows. In Section 2, we provide an overview of GAPs (past work), define a social network (SN for short), and explain how GAPs can represent some types of diffusion in SNs. Section 3 formally defines different types of social network diffusion optimization problems and provides results on their computational complexity and other properties. Section 4 shows how our framework can represent several existing diffusion models for social networks including economics and epidemiology. In Section 5 we present the exact SNDOP-Mon algorithm to answer SNDOP queries under certain assumptions of monotonicity. We then develop a greedy algorithm GREEDY-SNDOP and show that under certain conditions, it is guaranteed to be an $(\frac{e}{e-1})$ approximation algorithm for SNDOP queries — this is the best possible approximation guarantee. Last, but not least, we describe our prototype implementation and experiments in Section 6. Specifically, we tested our GREEDY-SNDOP algorithm on a real-world social network data set consisting of over 7000 nodes and over 103,000 edges from Wikipedia logs. We show that we solve social network diffusion optimization problems over real data sets in acceptable times. We emphasize that much additional work is required on further enhancing scalability and that research on social network diffusion optimization problems is at its very infancy. Finally, in Section 7, we review related work.

[3]Again, our framework allows us to add additional constraints — for instance, that medication can only be given to people satisfying certain conditions, e.g. be over a certain age, or be within a certain age range and not have any conditions that are contra-indicators for the medication in question.

2. TECHNICAL PRELIMINARIES

In this section, we first formalize social networks, then briefly review generalized annotated logic programs (GAPs) [Kifer and Subrahmanian 1992] and then describe how GAPs can be used to represent concepts related to diffusion in SNs.

2.1. Social Networks Formalized

Throughout this paper, we assume the existence of two arbitrary but fixed disjoint sets VP, EP of *vertex* and *edge predicate symbols* respectively. Each vertex predicate symbol has arity 1 and each edge predicate symbol has arity 2.

Definition 2.1. A **social network** is a 5-tuple $(\mathbf{V}, \mathbf{E}, \ell_{vert}, \ell_{edge}, w)$ where:

(1) $\mathbf{V}$ is a finite set whose elements are called *vertices*.
(2) $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$ is a finite multi-set whose elements are called *edges*.
(3) $\ell_{vert} : \mathbf{V} \rightarrow 2^{\mathsf{VP}}$ is a function, called *vertex labeling function*.
(4) $\ell_{edge} : \mathbf{E} \rightarrow \mathsf{EP}$ is a function, called *edge labeling function*. [4]
(5) $w : \mathbf{E} \rightarrow [0, 1]$ is a function, called *weight function*.

We now present a brief example of an SN.

Example 2.2. Let us return to the cell phone example (query **(Q1)**). Figure 1 shows a toy SN the cell phone company might use. Here, we might have $\mathsf{VP} = \{male, female, adopter, temp\_adopter, non\_adopter\}$ denoting the sex and past adoption behavior of each vertex; EP might be the set $\{phone, email, IM\}$ denoting the types of interactions between vertices (phone call, email, and instant messaging respectively). The function ℓ_{vert} is shown in Figure 1 by the shape (denoting past adoption status) and shading (male/female). The type of edges (bold for phone, dashed for email, dotted for IM) is used to depict ℓ_{edge}. $w(\langle v_1, v_2\rangle)$ denotes the percentage of communications of type $\ell_{edge}(\langle v_1, v_2\rangle)$ initiated by v_1 that were with v_2 (measured either w.r.t. time or bytes).

It is important to note that our definition of social networks is much broader than that used by several researchers [Anderson and May 1979; Coelho et al. 2008; Jackson and Yariv 2005; Kempe et al. 2003] who often do not consider either ℓ_{edge} or ℓ_{vert} or edge weights through the function w — it is well-known in marketing that intrinsic properties of vertices (customers, patients) and the nature and strength of the relationships (edges) is critical for decision making in those fields.

Note. We assume that SNs satisfy various integrity constraints. In Example 2.2, it is clear that $\ell_{vert}(v)$ should include at most one of $male, female$ and at most one of $adopter$, $temp\_adopter$,$non\_adopter$. *We assume the existence of some integrity constraints to ensure this kind of semantic integrity* – they can be written in any reasonable syntax to express ICs – in the rest of this paper, we assume that social networks have associated ICs and that they satisfy them. In our example, we will assume ICs ensuring that a vertex can be marked with at most one of $male, female$ and at most one of $adopter, temp\_adopter, non\_adopter$.

2.2. Generalized Annotated Programs: A Recap

We now recapitulate the definition of generalized annotated logic programs from [Kifer and Subrahmanian 1992]. We assume the existence of a set AVar of variable symbols ranging over the unit real interval $[0, 1]$ and a set $\mathcal{F}$ of function symbols each of which has an associated arity. We start by defining annotations.

[4] Each edge $e \in \mathbf{E}$ is labeled by exactly one predicate symbol from EP. However, there can be multiple edges between two vertices labeled with different predicate symbols.

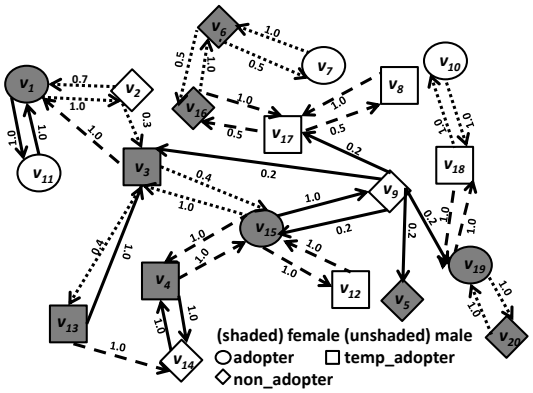


Fig. 1. Example cellular network.

Definition 2.3 (*Annotation*). Annotations are inductively defined as follows: (i) Any member of $[0, 1] \cup \mathsf{AVar}$ is an annotation.
(ii) If $f \in \mathcal{F}$ is an n-ary function symbol and $t_1, \ldots, t_n$ are annotations, then $f(t_1, \ldots, t_n)$ is an annotation.

For instance, $0.5, 1, 0.3$ and X are all annotations (here X is assumed to be a variable in AVar). If $+, *, /$ are all binary function symbols in $\mathcal{F}$, then $\frac{(X+1)*0.5}{0.3}$ is an annotation.[5]

We define a separate logical language whose constants are members of V and whose predicate symbols consist of $\mathsf{VP} \cup \mathsf{EP}$. We also assume the existence of a set $\mathcal{V}$ of variable symbols ranging over the constants (vertices). No function symbols are present. Terms and atoms are defined in the usual way (cf. [Lloyd 1987]). If $A = p(t_1, \ldots, t_n)$ is an atom and $p \in \mathsf{VP}$ (resp. $p \in \mathsf{EP}$), then A is called a *vertex* (resp. *edge*) atom. We will use $\mathcal{A}$ to denote the set of all ground atoms (i.e., atoms where no variable occurs).

Definition 2.4 (*annotated atom / GAP-rule / GAP*). If A is an atom and μ is an annotation, then $A : \mu$ is an *annotated atom*. If A is a vertex (resp. edge) atom, then $A : \mu$ is also called *vertex* (resp. *edge*) annotated atom. If $A_0 : \mu_0, A_1 : \mu_1, \ldots, A_n : \mu_n$ are annotated atoms, then

$$A_0 : \mu_0 \leftarrow A_1 : \mu_1 \wedge \ldots \wedge A_n : \mu_n$$

is called a *GAP rule* (or simply *rule*). When $n = 0$, the above rule is called a *fact*.[6] A *generalized annotated program* (GAP) is a finite set of rules. An annotated atom (resp. a rule, a GAP) is *ground* iff there are no occurrences of variables from either AVar or $\mathcal{V}$ in it.

[5]Notice that in [Kifer and Subrahmanian 1992] annotations are not restricted to be in $[0, 1]$ but any upper semi-lattice is allowed – for the purpose of this paper we will restrict ourselves to the unit real interval.
[6]For notational simplicity, we will often write a fact $A_0 : \mu_0 \leftarrow$ simply as $A_0 : \mu_0$, i.e. we drop the symbol $\leftarrow$.

Every social network $\mathcal{S} = (\mathbf{V}, \mathbf{E}, \ell_{vert}, \ell_{edge}, w)$ can be represented by the GAP $\Pi_{\mathcal{S}} = \{q(v) : 1 \leftarrow \mid v \in \mathbf{V} \wedge q \in \ell_{vert}(v)\} \cup \{ep(v_1, v_2) : w(\langle v_1, v_2\rangle) \leftarrow \mid \langle v_1, v_2\rangle \in \mathbf{E} \wedge \ell_{edge}(\langle v_1, v_2\rangle) = ep\}$.

Definition 2.5 (*embedded social network*). A social network $\mathcal{S}$ is said to be *embedded* in a GAP Π iff $\Pi_{\mathcal{S}} \subseteq \Pi$.

It is clear that all social networks can be represented as GAPs. When we augment $\Pi_{\mathcal{S}}$ with other rules — such as rules describing how certain properties diffuse through the social network, we get a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$ that captures both the structure of the SN and the diffusion principles. Here is a small example of such a GAP.

Example 2.6. The GAP Π_{cell} might consist of $\Pi_{\mathcal{S}}$ using the social network of Figure 1 plus the GAP-rules:

(1) $will\_adopt(V_0) : 0.8 \times X + 0.2 \leftarrow adopter(V_0) : 1 \wedge male(V_0) : 1 \wedge IM(V_0, V_1) : 0.3 \wedge female(V_1) : 1 \wedge will\_adopt(V_1) : X$.

(2) $will\_adopt(V_0) : 0.9 \times X + 0.1 \leftarrow adopter(V_0) : 1 \wedge male(V_0) : 1 \wedge IM(V_0, V_1) : 0.3 \wedge male(V_1) : 1 \wedge will\_adopt(V_1) : X$.

(3) $will\_adopt(V_0) : 1 \leftarrow temp\_adopter(V_0) : 1 \wedge male(V_0) : 1 \wedge email(V_1, V_0) : 1 \wedge female(V_1) : 1 \wedge will\_adopt(V_1) : 1$.

Rule (1) says that if V_0 is a male adopter and V_1 is female and the weight of V_0's instant messages to V_1 is 0.3 or more, and we previously thought that V_1 would be an adopter with confidence X, then we can infer that V_0 will adopt the new plan with confidence $0.8 \times X + 0.2$. The other rules may be similarly read.

Suppose $\mathcal{S}$ is a social network and $\Pi \supseteq \Pi_{\mathcal{S}}$ is a GAP. In this case, we call the rules in $\Pi - \Pi_{\mathcal{S}}$ *diffusion rules*. In this paper we consider a restricted class of GAPs: every rule with a non-empty body has a vertex annotated atom in the head ([Kifer and Subrahmanian 1992] allows any atom to appear in the head of a rule). Thus, edge atoms can appear only in rule bodies or facts. This means that neither edge weights nor edge labels change as the result of the diffusion. However, for the general case, it is possible for them to change as a result of the diffusion process.

GAPs have a formal semantics that can be immediately used. An interpretation I is any mapping from the set $\mathcal{A}$ of all grounds atoms to $[0, 1]$. The set $\mathcal{I}$ of all interpretations can be partially ordered via the ordering: $I_1 \preceq I_2$ iff for all ground atoms A, $I_1(A) \leq I_2(A)$. $\mathcal{I}$ forms a complete lattice under the $\preceq$ ordering.

Definition 2.7 (*satisfaction / entailment*). An interpretation I *satisfies* a ground annotated atom $A : \mu$, denoted $I \models A : \mu$, iff $I(A) \geq \mu$. I satisfies a ground GAP-rule r of the form $AA_0 \leftarrow AA_1 \wedge \ldots \wedge AA_n$ (denoted $I \models r$) iff either (i) I satisfies AA_0 or (ii) there exists an $1 \leq i \leq n$ such that I does not satisfy AA_i. I *satisfies* a non-ground annotated atom (rule) iff I satisfies all ground instances of it. I *satisfies* a GAP iff I satisfies all rules in it. A GAP Π *entails* an annotated atom AA, denoted $\Pi \models AA$, iff every interpretation I that satisfies Π also satisfies AA.

As shown by [Kifer and Subrahmanian 1992], we can associate a fixpoint operator with any GAP Π that maps interpretations to interpretations.

Definition 2.8. Suppose Π is any GAP and I an interpretation. The mapping $\mathbf{T}_\Pi$ that maps interpretations to interpretations is defined as $\mathbf{T}_\Pi(I)(A) = \mathbf{sup}\{\mu \mid A : \mu \leftarrow AA_1 \wedge \ldots \wedge AA_n$ is a ground instance of a rule in Π and for all $1 \leq i \leq n$, $I \models AA_i\}$.

[Kifer and Subrahmanian 1992] show that $\mathbf{T}_\Pi$ is monotonic (w.r.t. $\preceq$) and has a least fixpoint $lfp(\mathbf{T}_\Pi)$. Moreover, they show that Π entails $A : \mu$ iff $\mu \leq lfp(\mathbf{T}_\Pi)(A)$ and hence $lfp(\mathbf{T}_\Pi)$ precisely captures the ground atomic logical consequences of Π. They

also define the *iteration* of $\mathbf{T}_\Pi$ as follows: $\mathbf{T}_\Pi \uparrow 0$ is the interpretation that assigns 0 to all ground atoms; $\mathbf{T}_\Pi \uparrow (i+1) = \mathbf{T}_\Pi(\mathbf{T}_\Pi \uparrow i)$.

The semantics of GAPs requires that when there are multiple ground instances of GAP-rules with the same head that "fire", the highest annotation in any of these ground rules is "chosen" according to the semantics of GAPs. This might seem restrictive and counter-intuitive to some, but it actually is the source of much power of GAPs. For instance, one school of thought in probabilistic logic programming [Raedt et al. 2007] is that when multiple ground rules with the same head "fire", the annotation derived should be the "noisy-or" value derived by combining the values of the annotations in the heads of firing rules. However, this is just one way of combining evidence from multiple sources[7] - many other triangular co-norms other than noisy-or can be used and have been used in the literature [Bonissone 1987a]. However, such co-norms can be expressed in our framework. If we have ground rules $G_1, G_2, \ldots, G_n$, each having the same atom in the head, and we want to combine evidence using a triangular co-norm[8] $\oplus$, and if G_i has the form:

$$A : \mu_i \leftarrow Body_i$$

then we can replace these rules with the rules:

$$A : \oplus(\{\mu_i \mid i \in X\}) \leftarrow \bigwedge_{i \in X} Body_i$$

for any subset $X \subseteq \{1, \ldots, n\}$. Moreover, as we have already remarked, many real-world diffusion models are non-probabilistic, making assumptions about how annotations should be combined harder to justify. However, the above discussion shows that the GAP framework is capable of expressing such rules. Though there is clearly a cost in terms of difficulty of expressing such methods to combine evidence generated by multiple rules, algorithms already exist and have been implemented ([Broecheler et al. 2010]) to learn GAP-based diffusion rules automatically from social network time series data.

We will show (in Section 4) that many existing diffusion models for a variety of phenomena can be expressed as a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$ by adding some GAP-rules describing the diffusion process to $\Pi_{\mathcal{S}}$.

3. SOCIAL NETWORK DIFFUSION OPTIMIZATION PROBLEM (SNDOP) QUERIES

3.1. Basic SNDOP Queries

In this section, we develop a formal syntax and semantics for optimization in social networks, taking advantage of the aforementioned embedding of SNs into GAPs. In particular, we formally define SNDOP queries, examples of which have been informally introduced earlier as **(Q1)** and **(Q2)**. We see from queries **(Q1)** and **(Q2)** that a SNDOP query looks for a **set $\mathbf{V}'$** of vertices and has the following components: (i) an objective function expressed via an aggregate operator, (ii) an integer $k > 0$, (iii) a set of conditions that each vertex in $\mathbf{V}'$ must satisfy, (iv) an "input" atom $g_I(V)$, and (v) an "output" atom $g_O(V)$ (here g_I and g_O are vertex predicate symbols, whereas V is a variable).

[7]Thus far, we have not come across any real-world diffusion models that use noisy-OR combinations or indeed any triangular co-norm [Bonissone 1987a] other than the MAX used in this paper to combine values generated by multiple rules having the same head. However should such diffusion models come to light, it may be appropriate to explore the use of languages such as ProbLog to see if we can "do better" for those selected diffusion models.

[8]When we apply $\oplus$ to a set $\{x_1, \ldots, x_k\}$, we use $\oplus(\{x_1, \ldots, x_k\})$ as short-hand for $\oplus(x_1, \oplus(\{x_2, \ldots, x_n\}))$ which is well defined as all triangular co-norms are commutative and associative.

Aggregates. It is clear that in order to express queries like **(Q1)** and **(Q2)**, we need aggregate operators which are mappings $agg : \mathsf{FM}([0,1]) \rightarrow \mathbb{R}^+$ ($\mathbb{R}^+$ is the set of non-negative reals) where $\mathsf{FM}(X)$ denotes the set of all finite multisets that are subsets of X. Relational DB aggregates like SUM,COUNT,AVG,MIN,MAX are all aggregate operators which can take a finite multiset of non-negative reals as input and return a single non-negative real.

Vertex condition. A vertex condition is a set of vertex annotated atoms containing exactly one variable (intuitively, such annotated atoms are conditions that must be jointly satisfied by a vertex). More formally, a vertex condition VC is a set $\{p_1(V) : \mu_1, \ldots, p_n(V) : \mu_n\}$ where each $p_i \in \mathsf{VP}$, $V \in \mathcal{V}$, and each $\mu_i \in [0,1]$. We use $VC[V/v]$ to denote the set of ground annotated atoms obtained from VC by replacing each occurrence of V with v, that is $VC[V/v] = \{p_1(v) : \mu_1, \ldots, p_n(v) : \mu_n\}$. A GAP Π entails $VC[V/v]$, denoted $\Pi \models VC[V/v]$, iff $\Pi \models p_i(v) : \mu_i$ for all $1 \leq i \leq n$.

Thus, in our example, $\{male(V) : 1, adopter(V) : 1\}$ is a vertex condition, but $\{male(V) : 1, email(V, V') : 1\}$ is not. We are now ready to define a SNDOP query.

Definition 3.1 (*SNDOP query*). A *SNDOP query* is a 5-tuple $(agg, VC, k, g_I(V), g_O(V))$ where agg is an aggregate, VC is a vertex condition, $k > 0$ is an integer, and $g_I(V)$, $g_O(V)$ are vertex atoms.

Let us consider again the medication distribution plan example. Suppose we have a diffusion model expressing how a property *healthy* diffuses in a social network w.r.t. a property *immune* (which would hold for a vertex when a medication is given to it). An interesting query to pose would be to determine a set of at most k people such that if these people were immune to the disease, then the number of healthy people would be maximized. Such a query can be expressed with the SNDOP query $(\mathsf{SUM}, \emptyset, k, immune(V), healthy(V))$. Here, the goal is to find a set $\mathbf{V}' \subseteq \mathbf{V}$ of vertices such that $|\mathbf{V}'| \leq k$ and the following is maximized:

$$\mathsf{SUM}\{lfp(\mathbf{T}_{\Pi \cup \{immune(v') : 1 \mid v' \in \mathbf{V}'\}})(healthy(v)) \mid v \in \mathbf{V}\}$$

Here, the SUM is applied to a multiset rather than a set. Note that in the query above $VC = \emptyset$, meaning that the *immune* property can be assigned to any vertex of the SN. However, other queries can be expressed where VC imposes restrictions on which vertices can have property *immune*. As an example, $VC = \{adult(V)\}$ would enforce every vertex in $\mathbf{V}'$ to be an adult person.

If we return to our cell phone example, we can set $agg = \mathsf{SUM}$, $VC = \emptyset$, $k = 3$ (for example), $g_I(V) = will\_adopt(V)$, and $g_O(V) = will\_adopt(V)$ (notice that in this case $g_I(V) = g_O(V)$). Here also, the goal is to find a set $\mathbf{V}' \subseteq \mathbf{V}$ of vertices such that $|\mathbf{V}'| \leq 3$ and the following is maximized:

$$\mathsf{SUM}\{lfp(\mathbf{T}_{\Pi \cup \{will\_adopt(v') : 1 \mid v' \in \mathbf{V}'\}})(will\_adopt(v)) \mid v \in \mathbf{V}\}$$

Here, the SUM is applied to a multiset rather than a set. Note that the diffusion model's impact is captured via the $lfp(\mathbf{T}_{\Pi \cup \{will\_adopt(v') : 1 \mid v' \in \mathbf{V}'\}})(will\_adopt(v))$ expression which, intuitively, tells us the confidence (according to the diffusion model) that each vertex v will be an adopter. If we return to an extended version of our cell phone example and we want to ensure that the vertices in $\mathbf{V}'$ are "good" customers [9] then we merely can set $VC = \{good(V) : 1\}$. This query now asks us to find a set $\mathbf{V}'$ of three or less vertices — all of which are "good" customers of the company C — such that $\mathsf{SUM}\{lfp(\mathbf{T}_{\Pi \cup \{will\_adopt(v') : 1 \mid v' \in \mathbf{V}'\}})(will\_adopt(v)) \mid v \in \mathbf{V}\}$ is maximized.

[9] We can think of many ways a company may define "good" customers, e.g. those who regularly pay their bills on time, those who buy a lot of services from the company, those who have stayed as customers for a long time, etc. For our example, the specific definition of "good" is not relevant.

Our framework also allows the vertex condition VC to have annotations other than 1. So in our cell phone example, the company could explicitly exclude anyone whose "opinion" toward the company is negative. If opinion is quantified on a continuous $[0, 1]$ scale (such automated systems do exist [Subrahmanian and Recupero 2008]), then the vertex condition might be restated as $VC = \{good(V) : 1, negative\_opinion\_C(V) : 0.7\}$ which says that the company wants to exclude anyone whose negativity about the company exceeds 0.7 according to an opinion scoring engine such as [Subrahmanian and Recupero 2008].

Definition 3.2 (*pre-answer/value*). Consider a SN $\mathcal{S} = (\mathbf{V}, \mathbf{E}, \ell_{vert}, \ell_{edge}, w)$ embedded in a GAP Π. A *pre-answer* to the SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$ w.r.t. Π is any set $\mathbf{V}' \subseteq \mathbf{V}$ such that:

(1) $|\mathbf{V}'| \leq k$, and
(2) for all vertices $v'' \in \mathbf{V}'$, $\Pi \cup \{g_I(v') : 1 \mid v' \in \mathbf{V}'\} \models VC[V/v'']$.

We use $\mathsf{pre\_ans}(Q, \Pi)$ to denote the set of all pre-answers to Q w.r.t. Π (whenever Π is clear from the context we simply write $\mathsf{pre\_ans}(Q)$).

The *value* of a pre-answer $\mathbf{V}'$ is defined as follows:

$$value(\mathbf{V}') = agg(\{lfp(\mathbf{T}_{\Pi \cup \{g_I(v'):1 \mid v' \in \mathbf{V}'\}})(g_O(v)) \mid v \in \mathbf{V}\})$$

where the aggregate is applied to a multi-set rather than a set. We also note that we can define $value$ as a mapping from interpretations to reals based on a SNDOP query. We say $value(I) = agg(\{I(g_O(v)) \mid v \in \mathbf{V}\})$.

If we return to our cell phone example, $\mathbf{V}'$ is the set of vertices to which the company is considering giving free plans. $value(\mathbf{V}')$ is computed as follows.

(1) Find the least fixpoint of $\mathbf{T}_{\Pi'_{cell}}$ where Π'_{cell} is Π_{cell} expanded with facts of the form $will\_adopt(v') : 1$ for each vertex $v' \in \mathbf{V}'$.
(2) For each vertex $v \in \mathbf{V}$ (the entire set of vertices, not just $\mathbf{V}'$ now), we now find the confidence assigned by the least fixpoint.
(3) Summing up these confidences gives us a measure of the expected number of plan adoptees.

Definition 3.3 (*answer*). Suppose an SN $\mathcal{S} = (\mathbf{V}, \mathbf{E}, \ell_{vert}, \ell_{edge}, w)$ is embedded in a GAP Π and $Q = (agg, VC, k, g_I(V), g_O(V))$ is a SNDOP query. A pre-answer $\mathbf{V}'$ is an *answer* to the SNDOP query Q w.r.t. Π iff the SNDOP query has no other pre-answer $\mathbf{V}''$ such that $value(\mathbf{V}'') > value(\mathbf{V}')$.[10]

The *answer set* to SNDOP query Q w.r.t. Π, denoted $\mathsf{ans}(Q, \Pi)$, is the set of all answers to Q w.r.t. Π (whenever Π is clear from the context we simply write $\mathsf{ans}(Q)$).

It is important to note that an answer to an SNDOP query is a **set** of vertices that **jointly** maximize the objective function specified. Thus, it is entirely possible that if we set $k = 1$, we could have two answers $\{a_1\}$ and $\{a_2\}$ each of which ties for the highest value. However, $\{a_1, a_2\}$ may *not* be the answer that optimizes the objective function when $k = 2$.

Example 3.4. For instance, suppose a_1 and a_2 are brothers with largely the same connections. The sets $\{a_1\}$ and $\{a_2\}$ both have value 100 each and let us say these constitute an answer (looking at one individual only) w.r.t. an objective function, e.g. influencing voters in an election to vote for candidate X. As a_1, a_2 mostly influence the

[10] Throughout this paper, we only treat maximization problems - there is no loss of generality in this because minimizing an objective function f is the same as maximizing $-f$.

Table I. Special cases of SNDOPs

Type	Special Case	Reference
Special cases of Π	Linear GAP	Definition 3.6
Special cases of agg	Monotonic	Definition 3.7
	Positive-linear	Definition 3.8
Special cases of $value$	Zero-starting	Definition 3.10
	A-priori VC	Definition 3.12

same people, they may jointly be able to get only 110 people to vote for the candidate because of the large overlap in their sphere of influence. However, now consider persons a_3, a_4. Each of them can only influence 90 voters by themselves, but only 10 of these voters "overlap". Thus, they can jointly influence $80+80+10 = 170$ voters to vote for X. It would make more sense (all other things being equal) for the candidate's party to invest in $\{a_3, a_4\}$.

Example 3.5. Consider the GAP Π_{cell} of Example 2.6 with the social network from Figure 1 embedded and the SNDOP query $Q_{cell} = (SUM, \emptyset, 3, will\_adopt, will\_adopt)$. The sets $\mathbf{V}'_1 = \{v_{15}, v_{19}, v_6\}$ and $\mathbf{V}'_2 = \{v_{15}, v_{18}, v_6\}$ are both *pre-answers*. In the case of $\mathbf{V}'_1$, two applications of the $\mathbf{T}_\Pi$ operator yields a fixpoint where the vertex atoms formed with $will\_adopt$ and vertices in the set $\{v_{15}, v_{19}, v_6, v_{12}, v_{18}, v_7, v_{10}\}$ are annotated with 1. For $\mathbf{V}_2$, only one application of $\mathbf{T}_\Pi$ is required to reach a fixpoint. In the fixpoint, vertex atoms formed with $will\_adopt$ and vertices in the set $\{v_{15}, v_6, v_{12}, v_{18}, v_7, v_{10}\}$ are annotated with 1. As these are the only vertex atoms formed with $will\_adopt$ that have a non-zero annotation after reaching the fixed point, we know that $value(\mathbf{V}'_1) = 7$ and $value(\mathbf{V}'_2) = 6$.

3.2. Special Cases of SNDOPs

In this section, we examine several special cases of SNDOPs that still allow us to represent a wide variety of diffusion models. Table I illustrates the special cases discussed in this section while Table II illustrates various properties we prove (and the assumptions under which those properties are proved).

Special Cases of GAPs. First, we present a class of GAPs called *linear* GAPs. Intuitively, a GAP is linear if the annotations in the rule heads are linear functions and the annotations in the body are variables. It is important to note that a wide variety of diffusion models can be represented with GAPs that meet the requirements of this special case. We formally define linear GAPs below.

Definition 3.6 (*Linear GAP*). A GAP-rule is *linear* iff it is of the form:

$$H_0 : c_0 + c_1 \cdot X_1 + \cdots + c_n \cdot X_n \leftarrow A_1 : X_1 \wedge \ldots \wedge A_n : X_n$$

where each $c_i \in [0, 1]$, $\Sigma_{i=1}^n c_i \in [0, 1]$, and each X_j is a variable in AVar. A GAP is linear iff each rule in it is linear.

Note that linear GAPs allow for a wide variety of models to be expressed. Section 4 will show that several well-known network diffusion models can be embedded into our framework. Diffusion Models 4.2 and 4.4, reported in Section 4, are linear GAPs while Diffusion Models 4.1 and 4.3 are not.

Special Aggregates. We define two types of aggregates: *monotonic* aggregates and *positive-linear* aggregates.

To define monotonicity, we first define a partial order $\sqsubseteq$ on multi-sets of numbers as follows: given two multi-sets of numbers X_1 and X_2, we write $X_1 \sqsubseteq X_2$ iff there exists an *injective* mapping $\beta : X_1 \rightarrow X_2$ such that $\forall x_1 \in X_1, x_1 \leq \beta(x_1)$.

Definition 3.7 (*Monotonic Aggregate*). An aggregate agg is **monotonic** iff whenever $X_1 \sqsubseteq X_2$, it is the case that $agg(X_1) \leq agg(X_2)$.

Definition 3.8 (*Positive-Linear Aggregate*). An aggregate agg is **positive-linear** iff it is defined as follows: $agg(X) = c_0 + \Sigma_{x_i \in X}\ c_i \cdot x_i$, where X is a finite multiset and $c_i \geq 0$ for all $i > 0$.

In the previous definition, note that c_0 can be positive, negative, or 0. Thus, we only require that the coefficients associated with the elements of the multi-set be positive – we allow for an additive constant to be negative. One obvious example of a positive-linear aggregate is SUM. Moreover, any positive weighted sum will also meet these requirements.

PROPOSITION 3.9. *If agg is a positive-linear aggregate, then it is a monotonic aggregate.*

Special cases of the query. We now describe two special cases of the query: *zero-starting* and *a-priori VC* SNDOP queries. Intuitively, zero-starting means that $value(\emptyset) = 0$.

Definition 3.10 (*Zero-starting*). An SNDOP query is **zero-starting** w.r.t. a given social network $\mathcal{S}$ and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$ iff $value(\emptyset) = 0$.

Note that the function $value$ is *uniquely defined* by a social network, a SNDOP query, and a diffusion model Π and hence the above definition is well defined.

The following result states that if an SNDOP query Q with a positive-linear aggregate is not zero-starting, then we can always modify it into an "equivalent" SNDOP query Q' (i.e. $\mathsf{ans}(Q) = \mathsf{ans}(Q')$) which is zero-starting and still maintains a positive-linear aggregate.

PROPOSITION 3.11. *Let $Q = (agg, VC, k, g_I(V), g_O(V))$ be a SNDOP query which is not zero-starting w.r.t. a social network $\mathcal{S}$ and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$, and where agg is positive-linear. Let $agg'(X) = agg(X) - value(\emptyset)$. Then, $Q' = (agg', VC, k, g_I(V), g_O(V))$ is a SNDOP query which is zero-starting w.r.t. $\mathcal{S}$ and Π, $\mathsf{ans}(Q) = \mathsf{ans}(Q')$, and agg' is positive-linear.*

Recall that in order to check if a set of vertices $\mathbf{V}'$ is a pre-answer, we need to check for all vertices $v'' \in \mathbf{V}'$ if $\Pi \cup \{g_I(v') : 1 \mid v' \in \mathbf{V}'\} \models VC[V/v'']$ (see condition (2) of Definition 3.2). Intuitively, a SNDOP query has an *A-Priori VC* (w.r.t. a given social network $\mathcal{S}$ and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$) when we can check this condition by looking only at the original social network $\mathcal{S}$ (thereby disregarding Π), that is we can check for all vertices $v'' \in \mathbf{V}'$ if $\Pi_{\mathcal{S}} \cup \{g_I(v'') : 1\} \models VC[V/v'']$. We formally define this notion below.

Definition 3.12 (*A-Priori VC*). A SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$ has an **A-Priori VC** w.r.t. a given social network $\mathcal{S} = (\mathbf{V}, \mathbf{E}, \ell_{vert}, \ell_{edge}, w)$ and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$ iff for each $\mathbf{V}' \subseteq \mathbf{V}$ the following holds: for each $v'' \in \mathbf{V}'$, $\Pi \cup \{g_I(v') : 1 \mid v' \in \mathbf{V}'\} \models VC[V/v'']$ iff $\Pi_{\mathcal{S}} \cup \{g_I(v'') : 1\} \models VC[V/v'']$.

If, in the cell phone example, we require that the free cell phones are given to "good" vertices, then query **(Q1)** is a-priori VC because being "good" may be defined statically and is not determined by the diffusion process. Likewise, if we consider our medical example, in the case of an a-priori VC query **(Q2)** saying that an individual below 5 should not get the medicine, this boils down to a static labeling of each node's age (below 5 or not) which is not affected by the diffusion process.

Table II. Properties that can be proven given certain assumptions

Property	Assumptions
Monotonicity of $value$ (Lemma 3.13)	Monotonicity of agg
Multiset $\{\mathbf{V}' \subseteq \mathbf{V} \mid \mathbf{V}'$ *is a pre-answer*$\}$ is a uniform matroid (Lemma 3.14)	A-priori VC
Submodularity of $value$ (Theorem 3.15)	Linear GAP Positive-linear agg A-priori VC

Table III. How the various properties are leveraged in the Algorithms

Algorithm	Property
Exact algorithm with pruning (Section 5.2)	Monotonicity of $value$
Approx. Ratio on Greedy Algorithm (Section 5.3)	Submodularity Zero-starting Uniform matroid for the pre-answers

3.3. Properties of SNDOPs

In this section, we will prove several useful properties of SNDOPs that use various combinations of the assumptions presented in the previous section. Later, we will leverage some of these properties in our algorithms. Table II summarizes the different properties that we prove in this section (as well as what assumptions we make to prove these properties). Table III shows how these properties are leveraged in the algorithms that we will present later in the paper.

We say that function $value$ is monotonic iff $\mathbf{V}_1 \subseteq \mathbf{V}_2$ implies $value(\mathbf{V}_1) \leq value(\mathbf{V}_2)$ for any two sets of vertices $\mathbf{V}_1$ and $\mathbf{V}_2$. The first property we show is that the value function is monotonic if agg is monotonic.

LEMMA 3.13. *Given a SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$, a social network $\mathcal{S}$, and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$, if agg is monotonic (Definition 3.7), then $value$ (defined as per Q and Π) is monotonic.*

Before introducing the next result we recall the definitions of matroid and uniform matroid. A *matroid* is a pair (X, I) where X is a finite set and I is a collection of subsets of X (called "independent"), satisfying two axioms:

(1) $B \in I, A \subset B \Rightarrow A \in I$.
(2) $A, B \in I, |A| < |B| \Rightarrow \exists x \in B - A \; s.t. \; A \cup \{x\} \in I$.

A *uniform* matroid is a matroid such that independent sets are all sets of size at most k for some $k \geq 1$.

Next, we show that the set of pre-answers is a uniform matroid in the special case of an a-priori VC query.

LEMMA 3.14. *Given a SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$, a social network $\mathcal{S}$, and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$, if Q is a-priori VC w.r.t. $\mathcal{S}$ and Π, then the set of pre-answers is a uniform matroid.*

As we will see in Section 5, the above lemma (along with other properties, see Theorem 5.8) enables us to define a greedy approximation algorithm to solve SNDOP queries that achieves the best possible approximation ratio that a polynomial algorithm can achieve (unless $\mathbf{P} = \mathbf{NP}$).

An important property in social networks is *submodularity* whose relationship to the spread of phenomena in social networks has been extensively studied [Mossel and Roch 2007; Kleinberg 2008; Leskovec et al. 2007a]. If X is a set, then a function $f : 2^X \to \mathbb{R}$ is *submodular* iff whenever $X_1 \subseteq X_2 \subseteq X$ and $x \in X - X_2$, $f(X_1 \cup \{x\}) - f(X_1) \geq f(X_2 \cup \{x\}) - f(X_2)$. The following result states that the $value$ function associated with

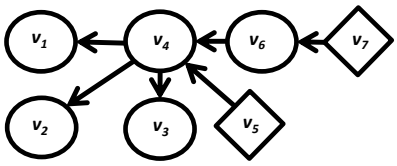


Fig. 2. Social network corresponding with Example 3.16 concerning disease spread.

a linear GAP and an a-priori VC SNDOP query whose aggregate is positive-linear is guaranteed to be submodular.

THEOREM 3.15. *Given an SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$, a social network $\mathcal{S}$, and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$, if the following criteria are met:*

— Π is a linear GAP,
— Q is a-priori VC, and
— agg is positive-linear,

*then $value$ (defined as per Q and Π) is **sub-modular**.*

In other words, for $\boldsymbol{V}_{cond} \equiv \{v' | v' \in \boldsymbol{V} \text{ and } (\Pi_{\mathcal{S}} \cup \{g_I(v') : 1\} \models VC[V/v'])\}$, if $\boldsymbol{V}_1 \subseteq \boldsymbol{V}_2 \subseteq \boldsymbol{V}_{cond}$ and $v \in \boldsymbol{V}_{cond} - \boldsymbol{V}_2$, then the following holds:

$$value(\boldsymbol{V}_1 \cup \{v\}) - value(\boldsymbol{V}_1) \geq value(\boldsymbol{V}_2 \cup \{v\}) - value(\boldsymbol{V}_2)$$

Proof Sketch: *Consider a linear polynomial with a variable for each vertex in the set of vertices that meet the a-priori VC, where setting the variable to 1 corresponds to the vertex being picked and setting it to 0 indicates otherwise. For any subset of vertices meeting the a-priori VC, there is an associated polynomial of this form such that when the variables corresponding to the vertices are set to 1 (and the rest set to 0), the answer is equal to the corresponding $value$ for that set. For a sets $\boldsymbol{V}_1, \boldsymbol{V}_2$ and vertex v (as per the statement), we show that submodualirty holds by manipulating such polynomials.*

Example 3.16. We now show an example of a SNDOP query and a non-linear GAP for which the $value$ function is **not** sub-modular. Figure 2 shows a social network with one edge predicate, *e* – all edges are weighted with 1. Nodes in the network are either susceptible to the disease (circles) or carriers (diamonds) - the associated predicates are *suc* and *car* respectively. Additionally, we have the predicates *inf*, *exp* denoting vertices that have been infected by or exposed to the disease. No vertex is initially exposed or infected in the social network of Figure 2.

Let $\Pi_{disease}$ be the embedding of this network plus the following diffusion rules.

$$\boldsymbol{exp}(V) : 1 \leftarrow \boldsymbol{inf}(V) : 1$$

$$\boldsymbol{exp}(V) : 1 \leftarrow \boldsymbol{e}(V', V) : 1 \wedge \boldsymbol{inf}(V') : 1 \wedge \boldsymbol{suc}(V) : 1$$

$$\boldsymbol{inf}(V) : \left\lfloor \frac{\sum_i I_i}{\sum_i E_i} \right\rfloor \leftarrow \boldsymbol{exp}(V) : 1 \wedge \bigwedge_{V_i | (V_i, V) \in \mathbf{E}} (edge(V_i, V) : E_i \wedge inf(V_i) : I_i)$$

Intuitively, the second rule says that a vertex becomes exposed if that vertex is susceptible and it has at least one incoming neighbor that is infected. The third rule states that a vertex becomes infected if it is exposed and all its incoming neighbors are infected.

Consider the function $value$ defined as per the SNDOP query $(SUM, \emptyset, 2, inf(V), inf(V))$ and $\Pi_{disease}$. Obviously, as the GAP is not linear, it

does not meet the requirements of Theorem 3.15 to prove submodularity. We can actually show through counterexample, that this SNDOP query is not submodular. Consider the following:

$$\textbf{\textit{value}}(\{v_1, v_5\}) - \textbf{\textit{value}}(\{v_1\}) = 1$$

(here $value(\{v_1, v_5\}) = 2$ and $value(\{v_1\}) = 1$) and

$$\textbf{\textit{value}}(\{v_1, v_7, v_5\}) - \textbf{\textit{value}}(\{v_1, v_7\}) = 4$$

(here $value(\{v_1, v_7, v_5\}) = 7$ and $value(\{v_1, v_7\}) = 3$).

This shows a clear violation of submodularity.

As an example of how the values above are determined, consider $value(\{v1, v5\})$. Notice that the third rule of $\Pi_{disease}$ is the only one that can be used to propagate the inf property, but in order for a vertex V to get infected using this rule, V has to be exposed first (and all its incoming neighbors have to be infected). When v_1 and v_5 are assumed to be infected, v_4 gets exposed (v_1 and v_5 get exposed as well because of the first rule). At this point, the exposed property cannot be propagated any further, and no vertex can get infected because no vertex is both exposed and has all its incoming neighbors infected (notice that v_4 cannot get infected because v_6 is not infected). Thus, $value(\{v_1, v_5\}) = 2$.

3.4. The Complexity of SNDOP Queries

We now study the complexity of answering an SNDOP query. First, we show that SNDOP query answering is NP-hard by a reduction from max k-cover [Feige 1998]. We show that the problem is NP-hard even when many of the special cases hold.

THEOREM 3.17. *Finding an answer to an SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$ (w.r.t. a social network $\mathcal{S}$ and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$) is NP-hard (even if Π is a linear GAP, $VC = \emptyset$, $agg = SUM$ and $value$ is zero-starting).*

Proof Sketch: *The known NP-hard problem of MAX-K-COVER [Feige 1998] is defined as follows.*
INPUT: Set of elements, S and a family of subsets of S, $\mathcal{H} \equiv \{H_1, \ldots, H_{max}\}$, and positive integer K.
OUTPUT: Less than or equal to K subsets from $\mathcal{H}$ such that the union of the subsets covers a maximal number of elements in S.
We show that MAX-K-COVER can be embedded into a social network and that the corresponding SNDOP query gives an optimal answer to MAX-K-COVER. The embedding is done by creating a social network resembling a bipartite graph, where vertices represent either the elements or the subsets from the input of MAX-K-COVER. For every vertex pair representing a set and an element of that set, there is an edge from the set vertex to the element vertex. A single vertex and edge predicate are used - $vertex$ and $edge$. A single non-ground diffusion rule is added to the GAP: $vertex(V) : X \leftarrow vertex(V') : X \wedge edge(V', V) : 1$. The aggregate is simply the sum of the annotations associated with the vertex atoms. We show that the picked vertices that maximize the aggregate correspond with picked subsets that maximize output of the problem. Also, as we do not use VC, the GAP is linear, and the aggregate is positive-linear, we know that the $value$ function is submodular.

Under some conditions, the decision problem for SNDOP queries is also in NP.

THEOREM 3.18. *Given a SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$, a social network $\mathcal{S}$, a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$, and a real number $target$, the problem of checking whether*

there exists a pre-answer $\mathbf{V}'$ *s.t.* $value(\mathbf{V}') \geq target$ *is in NP under the assumptions that* agg *and the functions in* $\mathcal{F}$ *are polynomially computable, and* Π *is ground.*

Most common aggregate functions like SUM, AVERAGE, Weighted average, MIN, MAX, COUNT are all polynomially computable. Moreover, the assumption that the functions corresponding to the function symbols in $\mathcal{F}$ (i.e. the function symbols that can appear in the annotations of a GAP) are polynomially computable is also reasonable.

Later in this paper, we shall address the problem of answering a SNDOP query using an approximation algorithm. We re-state the definition of approximation below (see [Garey and Johnson 1979]).

Definition 3.19 (*Approximation*). Consider a maximization problem and let $OPT(I)$ denote the value of an optimal solution for an instance I of the problem. An α-approximation algorithm A is an algorithm that for any instance I finds a candidate solution such that

$$OPT(I) \leq \alpha \cdot A(I)$$

where $A(I)$ denotes the value of the solution found by A for instance I.

Based on the above definition, we shall say that $\mathbf{V}'$ is a $\frac{1}{\alpha}$-approximation to an SNDOP query if $value(\mathbf{V}_{opt}) \leq \alpha \cdot value(\mathbf{V}')$ (where $\mathbf{V}_{opt}$ is an answer to the SNDOP query). Likewise, the algorithm that produces $\mathbf{V}'$ in this case is an α-approximation algorithm. We note that due to the nature of the reduction from MAX-K-COVER that we used to prove NP-hardness, we can now show that unless $\mathbf{P} = \mathbf{NP}$, there is no PTIME-approximation of the SNDOP query answering problem that can guarantee that the approximate answer is better than 0.63 of the optimal value. This gives us an idea of the limits of approximation possible for a SNDOP query with a polynomial-time algorithm. Later, we will develop a greedy algorithm that precisely matches this approximation ratio.

THEOREM 3.20. *Answering a SNDOP query* $Q = (agg, VC, k, g_I(V), g_O(V))$ *(w.r.t. a social network* $\mathcal{S}$ *and a GAP* $\Pi \supseteq \Pi_{\mathcal{S}}$*) cannot be approximated in PTIME within a ratio of* $\frac{e-1}{e} + \epsilon$ *for some* $\epsilon > 0$ *(where* e *is the inverse of the natural log) unless* $\mathbf{P} = \mathbf{NP}$ *– even if* Π *is a linear GAP,* $VC = \emptyset$*,* $agg = SUM$ *and* $value$ *is zero-starting.*

In other words, the previous theorem says that there is no polynomial-time algorithm that can approximate $value$ within a factor of about 0.63 under standard assumptions.

3.5. Counting Complexity of SNDOP-Queries

In this section, we ask the question: how many answers are there to a SNDOP query $(agg, VC, k, g_I(V), g_O(V))$? In the case of the cell phone example, this corresponds to asking "How many sets ANS of people are there in the the network such that ANS has k or fewer people and ANS optimizes the aggregate, subject to the vertex condition VC?" If there are m such sets $ANS_1, \ldots, ANS_m$, this means the cell phone company can give the free cell phone plan to either all members of ANS_1 or to all members of ANS_2, and so forth. The "counting complexity" problem of determining m is is $\#P$-complete.

THEOREM 3.21. *Counting the number of answers to a SNDOP query* Q *(w.r.t. a social network* $\mathcal{S}$ *and a GAP* $\Pi \supseteq \Pi_{\mathcal{S}}$*) is #P-complete.*

3.6. The SNDOP-ALL Problem

Suppose the cell phone company wants to identify all the "most influential" users, that is, the users that when considered *individually* (and not as a set) yield a maximum

expected number of plan adoptees. This might be computed by taking the union of all the answers to query **(Q1)** with $k = 1$. For instance, if we consider the hypothetical example of the political candidate (Example 3.4), the candidate may also want to know all the top influencers when considered *individually*. In this case, vertices a_1, a_2 would emerge in the answer to a SNDOP-ALL query defined below.

Although the counting version of the query is $\#P$-hard, finding the *union* of all answers to a SNDOP query is no harder than a SNDOP query (w.r.t. polynomial-time Turing reductions). We shall refer to this problem as *SNDOP-ALL* - and it reduces both to and from a regular SNDOP query (w.r.t. polynomial-time Turing reductions).

We start with the following result, showing that we can answer a SNDOP query in PTIME with an oracle to SNDOP-ALL.

THEOREM 3.22. *Given a SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$, a social network $\mathcal{S}$, and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$), there exists a polynomial-time algorithm with an oracle to SNDOP-ALL which answers Q.*

Proof Sketch: *We embed a SNDOP query in a SNDOP-ALL query via the following informal algorithm (FIND-SET) that takes an instance of SNDOP-ALL (Q) and some vertex set V^*, $|V^*| \leq k$.*

(1) If $|V^*| = k$, return V^*
(2) Else, solve SNDOP-ALL(V^*), returning set V''.
  (a) If $V'' - V^* \equiv \emptyset$, return V^*
  (b) Else, pick $v \in V'' - V^*$ and return FIND-SET$(Q, V^* \cup v)$

The theorem below shows that SNDOP-ALL can be answered in PTIME with an oracle to a SNDOP query.

THEOREM 3.23. *Given a SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$, a social network $\mathcal{S}$, and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$), finding $\bigcup_{\boldsymbol{V'} \in ans(Q)} \boldsymbol{V'}$ reduces to $|\boldsymbol{V}| + 1$ SNDOP queries, where $\boldsymbol{V}$ is the set of vertices of $\mathcal{S}$.*

Proof Sketch: *Using an oracle that correctly answers SNDOP queries, we can answer a SNDOP-ALL query by setting up $|\boldsymbol{V}|$ SNDOP queries as follows:*

— Let k_{all} be the k value for the SNDOP-ALL query and for each SNDOP query i, let k_i be the k for that query. For each query i, set $k_i = k_{all} - 1$.
— Number each element of $v_i \in \boldsymbol{V}$ such that $g_I(v_i)$ and $VC(v_i)$ are true. For the ith SNDOP query, let v_i be the corresponding element of $\boldsymbol{V}$
— Let Π_i refer to the GAP associated with the ith SNDOP query and Π_{all} be the program for SNDOP-ALL. For each program Π_i, add fact $g_I(v_i) : 1$
— For each SNDOP query i, the remainder of the input is the same as for SNDOP-ALL.

After the construction, do the following:

(1) We shall refer to a SNDOP query that has the same input as SNDOP-ALL as the "primary query." Let $\boldsymbol{V'}_{ans}{}^{(pri)}$ be an answer to this query and $value(\boldsymbol{V'}_{ans}{}^{(pri)})$ be the associated value.
(2) For each SNDOP query i, let $\boldsymbol{V'}_{ans}{}^{(i)}$ be an answer and $value(\boldsymbol{V'}_{ans}{}^{(i)})$ be the associated value.
(3) Let $\boldsymbol{V'}$, the solution to SNDOP-ALL be initialized as $\emptyset$.
(4) For each SNDOP query i, if $value(\boldsymbol{V'}_{ans}{}^{(i)}) = value(\boldsymbol{V'}_{ans}{}^{(pri)})$, then add vertex v_i to $\boldsymbol{V'}$.

4. APPLYING SNDOPS TO REAL DIFFUSION PROBLEMS

In this section, we show how SNDOPs can be applied to real-word diffusion problems. Most diffusion models in the literature fall into one of three categories – **tipping** models (Section 4.1), where a given vertex adopts a behavior based on the ratio of how many of its neighbors previously adopted the behavior, **cascade** models (Section 4.2), where a property passes from vertex to vertex solely based on the strength of the relationship between the vertices, and **homophilic** models (Section 4.3), where vertices with similar properties tend to adopt the same behavior – irrespective (or in addition to) of network relationships. *None of these approaches solves SNDOP queries — they merely specify diffusion models rather than performing the kinds of optimizations that we perform in SNDOP queries.*

4.1. Tipping Diffusion Models

Tipping models [Centola 2010; Schelling 1978; Granovetter 1978] have been studied extensively in economics and sociology to understand diffusion phenomena. In tipping models, a vertex changes a property based on the cumulative effect of its neighbors. In this section, we present the tipping model of Jackon-Yariv [Jackson and Yariv 2005], which generalizes many existing tipping models.

The Jackson-Yariv Diffusion Model [Jackson and Yariv 2005]. In this framework, the social network is just an undirected graph $\mathbf{G}' = (\mathbf{V}', \mathbf{E}')$ consisting of a set of agents (e.g. people). Each agent has a default behavior (A) and a new behavior (B). Suppose d_i denotes the degree of a vertex v_i. [Jackson and Yariv 2005] use a function $\gamma : \{0, \ldots, |\mathbf{V}'| - 1\} \rightarrow [0, 1]$ to describe how the number of neighbors of v affects the benefits to v for adopting behavior B. For instance, $\gamma(3)$ specifies the benefits (in adopting behavior B) that accrue to an arbitrary vertex $v \in \mathbf{V}'$ that has three neighbors. Let π_i denote the fraction of neighbors of v_i that have adopted behavior B. Let constants b_i and ρ_i be the agent-specific benefit and cost, respectively, for vertex v_i to adopt behavior B. [Jackson and Yariv 2005] state that node v_i switches to behavior B iff $\frac{b_i}{\rho_i} \cdot \gamma(d_i) \cdot \pi_i \geq 1$.

Returning to our cell-phone example, one could potentially use this model to describe the spread of the new plan. In this case, behavior A would be adherence to the current plan the user subscribes to, while B would be the use of the new plan. The associated SNDOP query would find a set of nodes which, if given a free plan, would jointly maximize the expected number of adoptees of the plan. Cost and benefit could be computed from factors such as income, time invested in switching plans, etc. We show how the model of [Jackson and Yariv 2005] can be expressed via GAPs.

DIFFUSION MODEL 4.1 (JACKSON-YARIV MODEL). *Given a Jackson-Yariv model consisting of $\mathbf{G}' = (\mathbf{V}', \mathbf{E}')$, we can set up a social network $\mathcal{S} = (\mathbf{V}', \mathbf{E}'', \ell_{vert}, \ell_{edge}, w)$ as follows. We define $\mathbf{E}'' = \{(x, y), (y, x) \mid (x, y) \in \mathbf{E}'\}$. We have a single edge predicate symbol $edge$ which is assigned by ℓ_{edge} to every edge in $\mathbf{E}''$, and w assigns 1 to all edges in $\mathbf{E}''$. Our associated GAP Π_{JY} now consists of $\Pi_{\mathcal{S}}$ plus one rule of the following form for each vertex v_i:*

$$B(v_i) : \left\lfloor \frac{b_i}{\rho_i} \cdot \gamma \left(\sum_j E_j \right) \cdot \frac{\sum_j X_j}{\sum_j E_j} \right\rfloor \leftarrow \bigwedge_{v_j \mid \langle v_j, v_i \rangle \in \mathbf{E}''} (edge(v_j, v_i) : E_j \wedge B(v_j) : X_j)$$

It is easy to see that this rule (when applied in conjunction with $\Pi_{\mathcal{S}}$ for a social network $\mathcal{S}$) precisely encodes the Jackson-Yariv semantics.

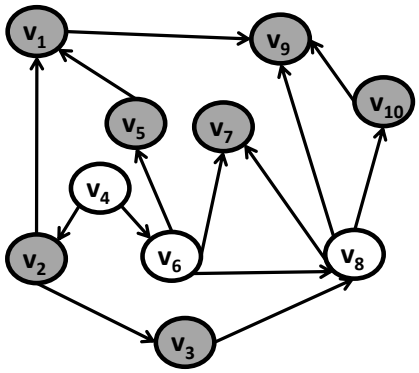


Fig. 3. Social network of individuals sharing photographs.

We notice right away that the above GAP is not linear. However, if we eliminate the floor function and impose certain restrictions on the coefficients appearing in the head of the rules, then we obtain a linear GAP that represents a variant of this model where the annotation would represent a "confidence" that an agent adopts behavior B. The idea of the confidence of an agent represented by a vertex adopting a certain behavior as a function of his adopting neighbors is suggested in the experiment of [Centola 2010] where he observed that the preference for a new behavior increases monotonically with the number of incoming neighbors who previously adopted said behavior. Such a linear GAP model is presented below.

DIFFUSION MODEL 4.2 (LINEAR JACKSON-YARIV MODEL). *For each vertex v_i let*

$$c_i = \frac{b_i}{\rho_i} \cdot \gamma \left(\sum_{v_j | \langle v_j, v_i \rangle \in \boldsymbol{E}'} w(\langle v_j, v_i \rangle) \right) \cdot \frac{1}{\sum_{v_j | \langle v_j, v_i \rangle \in \boldsymbol{E}'} w(\langle v_j, v_i \rangle)}$$

If for each vertex v_i, $c_i \in [0,1]$ and $|\{v_j \mid \langle v_j, v_i \rangle \in \boldsymbol{E}'\}| \times c_i \leq 1$, then we can derive a linear GAP for the Jackson-Yariv model that consists of one rule of the following form for each vertex v_i

$$B(v_i) : c_i \sum_j X_j \leftarrow \bigwedge_{v_j | \langle v_j, v_i \rangle \in \boldsymbol{E}'} B(v_j) : X_j$$

Notice that the above rule is similar to the one in Diffusion Model 4.1, but the floor function has been dropped and restrictions on the c_i's are imposed to make the rule linear.

Example 4.1. Figure 3 illustrates a social network of individuals who share photographs. Each edge is labeled with the predicate $share$ and has weight 1. The only vertex predicate we consider in this case is $buys\_camera$.

A vendor wishes to sell a camera and wants to see how the popularity of the camera will spread in the network. He wants to use a Jackson-Yariv style diffusion model. Suppose the social network is embedded into GAP Π which has one Jackon-Yariv style

Table IV. Comparison between standard and linear Jackson-Yariv Models

Vertex Atom	Annotation Assigned by $lfp(\mathbf{T}_{\Pi_{camera} \cup \{buys\_camera(v_2):1\}})$	Annotation Assigned by $lfp(\mathbf{T}_{\Pi_{lin} \cup \{buys\_camera(v_2):1\}})$
$buys\_camera(v_1)$	0.0	0.5
$buys\_camera(v_2)$	1.0	1.0
$buys\_camera(v_3)$	1.0	1.0
$buys\_camera(v_4)$	0.0	0.0
$buys\_camera(v_5)$	0.0	0.0
$buys\_camera(v_6)$	0.0	0.0
$buys\_camera(v_7)$	0.0	0.25
$buys\_camera(v_8)$	0.0	0.5
$buys\_camera(v_9)$	0.0	0.5
$buys\_camera(v_{10})$	0.0	0.5
SUM	2	4.25

tipping diffusion rule of the following form for each vertex v:

$$buys\_camera(v) : \left\lfloor \frac{\sum_j X_j}{\sum_j E_j} \right\rfloor \leftarrow \bigwedge_{v_j | \langle v_j, v \rangle \in \mathbf{E}} (shares(v_j, v) : E_j \wedge buys\_camera(v_j) : X_j)$$

We will call the GAP with the above diffusion rule $\Pi_{standard}$. Alternatively, we could have a linear version of it as follows:

$$buys\_camera(v) : \frac{\sum_j X_j}{|\{v_j | \langle v_j, v \rangle\}|} \leftarrow \bigwedge_{v_j | \langle v_j, v \rangle \in \mathbf{E}} buys\_camera(v_j) : X_j$$

We will call the GAP formed with the previous kind of diffusion rules Π_{lin}. In this case, it is clear that each rule head is annotated with the linear expression:

$$c_o + c_1 \cdot X_1 + \ldots + c_{|\{v_j | (v_j, v) \in \mathbf{E}\}|} \cdot X_{|\{v_j | (v_j, v) \in \mathbf{E}\}|}$$

Here, $c_0 = 0$ and for all $i > 0$ we have,

$$c_i = \frac{1}{|\{v_j | (v_j, v) \in \mathbf{E}\}|}$$

Clearly, each $c_i \in [0, 1]$ and the sum of all these constants is 1, which gives us linearity in accordance with Definition 3.6. Table IV shows the least fixed point for the two different GAPs (original JY model and the linear version) that arise when we assign an annotation of 1 to vertex atom $buys\_camera(v_2)$ — it also shows the sum of the annotations.

4.2. Cascading Diffusion Models

In a **cascading** model, a vertex obtains a property from one of its neighbors, typically based on the strength of its relationship with the neighbor. These models were introduced in the epidemiology literature in the early 20th century, but gained increased notice with the seminal work of [Anderson and May 1979]. Recently, cascading diffusion models have been applied to other domains as well. For example, [Cha et al. 2008] (diffusion of photos in Flickr) and [Sun et al. 2009] (diffusion of bookmarks in FaceBook) both look at diffusion process in social networks as "social cascades" of this type. In this section, we present an encoding of the popular SIR model of disease spread in our framework.

The SIR Model of Disease Spread. The SIR (*susceptible, infectious, removed*) model of disease spread [Anderson and May 1979] is a classic disease model which labels each

vertex in a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ (of humans) with *susceptible* if it has not had the disease but can receive it from one of its neighbors, *infectious* if it has caught the disease and t_{rec} units of time have not expired, and *removed* when the vertex can no longer catch or transmit the disease. The SIR model assumes that a vertex v that is infected can transmit the disease to any of its neighbors v' with a probability $p_{v,v'}$ for t_{rec} units of time. It is assumed that becoming infected takes precisely a time unit. We would like to find a set of at most k vertices that would maximize the expected number of vertices that become infected. These are obviously good candidates to treat with appropriate medications.

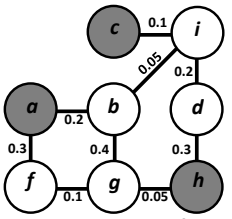
DIFFUSION MODEL 4.3 (SIR MODEL). *Let $\mathcal{S} = (\boldsymbol{V}, \boldsymbol{E}, \ell_{vert}, \ell_{edge}, w)$ be an SN where each edge is labeled with the predicate symbol e and $w(\langle v, v' \rangle) = p_{v,v'}$ assigns a probability of transmission to each edge . We use the predicate inf to designate the initially infected vertices. In order to create a GAP Π_{SIR} capturing the SIR model of disease spread, we first define t_{rec} predicate symbols $rec_1, \ldots, rec_{t_{rec}}$ where $rec_i(v)$ intuitively means that node v was infected i days ago. Hence, $rec_{t_{rec}}(v)$ means that v is "removed." We embed $\mathcal{S}$ into GAP Π_{SIR} by adding the following diffusion rules. If $t_{rec} > 1$, we add a non-ground rule for each $i = \{2, \ldots, t_{rec}\}$ - starting with t_{rec}:*

$$\begin{aligned} \boldsymbol{rec}_i(V) : R \;&\leftarrow\; \boldsymbol{rec}_{i-1}(V) : R \\ \boldsymbol{rec}_1(V) : R \;&\leftarrow\; \boldsymbol{inf}(V) : R \\ \boldsymbol{inf}(V) : (1-R) \cdot P_{V',V} \cdot P_{V'} \cdot (1-R') \;&\leftarrow\; \boldsymbol{rec}_{t_{rec}}(V) : R \wedge \boldsymbol{e}(V', V) : P_{V',V} \;\wedge \\ &\qquad \boldsymbol{inf}(V') : P_{V'} \wedge \boldsymbol{rec}_{t_{rec}}(V') : R'. \end{aligned}$$

The first rule says that if a vertex is in its $(i-1)$'th day of recovery with confidence R in the j'th iteration of the $\mathbf{T}_{\Pi_{SIR}}$ operator, then the vertex is i days into recovery (with the same confidence) in the $j+1$'th iteration of the operator. Likewise, the second rule intuitively encodes the fact that if a vertex became infected with confidence R in the j'th iteration of the $\mathbf{T}_{\Pi_{SIR}}$ operator, then the vertex is one day into recovery in the $j+1$'th iteration of the operator with the same confidence. The last rule says that if a vertex V' was infected with confidence $P_{V'}$ and has not been removed with confidence $1-R'$, and there is an edge from V' to V in the social network (weighted with $P_{V',V}$), given the confidence $1-R$ that V has not already been removed, then the confidence that the vertex V gets infected is $(1-R) \cdot P_{V',V} \cdot P_{V'}(1-R')$. Here, $P_{V'}(1-R')$ is one way of measuring the confidence that V' is infected and has not recovered and $P_{V',V}$ is the confidence of infecting the neighbor. Notice that e is a static property of the graph which does not change over the time, so we do not need time indexes for it. As for inf, the reason why we can avoid using time indexes is that we can keep track of how much time has gone since a vertex got infected with the predicates rec_i using the second rule.

To see how this GAP works, we execute a few iterations of the $\mathbf{T}_{\Pi_{SIR}}$ operator and show the fixpoint that it reaches on the small graph shown in Figure 4. In this graph, the initial infected vertices are those shown as a shaded circle. The transmission probabilities weight the edges in the graph.

The SNDOP query $(SUM, \emptyset, k, inf, inf)$ can be used to compute the *expected number* of infected vertices in the least fixpoint of T_Π. This query says "find a set of at most k vertices in the social network which, if infected, would cause the maximal number of vertices to become infected in the future." However, the above set of rules can be easily used to express other things. For instance, an epidemiologist may not be satisfied with only one set of k vertices that can cause the disease to spread to the maximum extent - as there may be another, disjoint set of k vertices that could cause the same effect. The epidemiologist may want to find all members of the population, that if present in a group of size k could spread the disease to a max-



1	inf(a):1, inf(c):1, inf(h):1
2	rec_1(a):1, rec_1(c):1, rec_1(h):1, inf(b):0.2, inf(d):0.3, inf(f):0.3, inf(g):0.05, inf(i):0.1
3	rec_2(a):1, rec_2(c):1, rec_2(h):1, rec_1(b):0.2, rec_1(d):0.3, rec_1(f):0.3, rec_1(g):0.05, rec_1(i):0.1 inf(g):0.08
4	rec_2(b):0.2, rec_2(d):0.3, rec_2(f):0.3, rec_2(g):0.05, rec_2(i):0.1, rec_1(g):0.08
5	rec_2(g):0.08

Fig. 4. Left: Sample network for disease spread. Right: annotated atoms entailed after each application of $\mathbf{T}_{\Pi_{SIR}}$ (maximum, non-zero annotations only).

imum extent. This can be answered using a *SNDOP-ALL* query, described in Section 3.

The SIS Model of Disease Spread. The SIS (Susceptible-Infectious-Susceptible) model [Hethcote 1976] is a variant of the SIR model where an individual becomes susceptible to disease after recovering (as opposed to SIR, where an individual acquires permanent immunity). SIS can be easily represented by a simple modification to the SIR model.

DIFFUSION MODEL 4.4 (SIS MODEL). *Take Diffusion Model 4.3 and change the third rule to*

$$inf(V) : P_{V',V} \cdot P_{V'} \cdot (1 - R') \leftarrow e(V', V) : P_{V',V} \wedge inf(V') : P_{V'} \wedge rec_{t_{rec}}(V') : R'.$$

Here, we do not consider the probability that vertex V is immune – hence the probability of recovery does not change the probability of becoming infected.

Diffusion in the Flickr Photo Sharing Network. The Flickr social network allows users to share photographs. Users create a list of "favorite" photos that can be viewed by other users. [Cha et al. 2008] use a variant of SIS above to study how photographs spread to the favorite lists of different users. A key difference is that they do not consider a node "recovered" – i.e. once a photo was placed on a favorite list, it was relatively permanent (the study was conducted over about 100 days). They also found that photos lower on a favorite list (as the result of a user marking a large number of photos as "favorite") for a given user could still spread through the network. A simple GAP that captures the intuition of how Flickr photos spread according to [Cha et al. 2008] uses just one rule:

DIFFUSION MODEL 4.5 (FLICKR PHOTO DIFFUSION).

$$photo_i(V) : const_i \cdot X_i \leftarrow connected\_to(V', V) : 1 \wedge photo_i(V') : X_i$$

In Diffusion Model 4.5, the annotation of the vertex atom $photo_i(V)$ is the confidence that vertex V has marked photo i as one of its favorites. The predicate *connected_to* is the sole edge label representing that there is a connection from vertex V' to V (users select other users on this network). Additionally, the value $const_i$ is a number in $[0, 1]$ that determines how a given photo spreads in the network. Notice that the above rule is linear, as the head is a linear combination and $const_i \in [0, 1]$. We note that for all of these models, the annotation functions reflect one interpretation of the confidence that a vertex is infected or recovered – others are possible in our framework.

4.3. Homophilic Diffusion Models

Recently, [Aral et al. 2009] studied the spread of mobile application use on a global instant-messaging network of over 27 million vertices. They found that network-based diffusion could overestimate the spread of a mobile application and, for this scenario, over 50% of the adopted use of the applications was due to **homophily** - vertices with similar properties adopting similar applications. Further, the recent experiment of [Centola 2011] illustrates that homophily plays a role in enhancing adoption under the tipping model.

These results should not be surprising – the basic idea behind web-search advertising is that two users with a similar property (search term) will be interested in the same advertised item. In fact, [Cha et al. 2008] explicitly pre-processed their Flickr data set with a heuristic to *eliminate* properties attached to vertices that could not be accounted for by a diffusion process. We can easily represent homophilic diffusion in a GAP with the following type of diffusion rule:

DIFFUSION MODEL 4.6 (HOMOPHILIC DIFFUSION OF A PRODUCT).

$$\textit{buys\_product}(V) : 0.5 \times X \leftarrow \textit{property}(V) : 1 \wedge \textit{exposed\_to\_product}(V) : X$$

In Diffusion Model 4.6, if a vertex is exposed to a product (e.g. through mass advertising) and has a certain property, then the person associated with the vertex purchases the product with a confidence of $0.5 \times X$, where X measures the extent of the exposure. For this rule, there are no network effects.

In [Watts and Peretti 2007], the authors propose a "big seed" marketing approach that combines both homophilic and network effects. They outline a strategy of advertising to a large group of individuals who are likely to spread the advertisement further through network effects. We now describe a GAP that captures the ideas underlying big seed marketing. Suppose we have a set of vertex predicate symbols $\mathsf{AL} \subseteq \mathsf{VP}$ corresponding to people "attributes" – these may be certain demographic characteristics such as education level, race, level of physical fitness, etc.. Suppose we want to advertise to people having (at least) one of $k \leq |\mathsf{AL}|$ attributes to maximize an aggregate agg with respect to a goal predicate g (in other words, we want to choose k attributes and advertise to people having those attributes so that agg with respect to g is maximized). Consider the following construction.

DIFFUSION MODEL 4.7 (BIG SEED MARKETING). *The GAP includes an embedding of the social network as well as the network diffusion model of the user's choice. We make the the following additions to the GAP and the SN:*

(*1*) *Add vertex predicate symbol* $attrib$ *to* VP.
(*2*) *For each* $lbl \in \mathsf{AL}$*, add a vertex* v_{lbl} *to* $\mathbf{V}$*. We also set* $\ell_{vert}(v_{lbl}) = \{attrib\}$.
(*3*) *For each* $lbl \in \mathsf{AL}$*, add the following non-ground rule:*

$$\boldsymbol{g}(V) : \mathit{eff}_{lbl} \times X \leftarrow lbl(V) : 1 \wedge \boldsymbol{g}(v_{lbl}) : X$$

where eff_{lbl} *is a constant in* $[0, 1]$ *corresponding to the confidence that, if advertised to, a vertex* v *with attribute* lbl *obtains an annotation of* 1 *on* $g(v)$.

Our SNDOP query is $(agg, VC, k, g(V), g(V))$*, where* $VC = \{attrib(V) : 1\}$.

Note that in the above diffusion model, the v_{lbl} vertices correspond to advertisements directed toward different vertex properties. The VC condition forces the query to only return v_{lbl} vertices. As an example, a solution like $\{g(v_{lbl_1}), g(v_{lbl_2})\}$ means that we are targeting people having attribute lbl_1 or lbl_2. The diffusion rule, added per label, ensures that the mass advertisement is received and that the vertex acts accordingly (hence the eff_{lbl} constants).

We close this section with a note that while all diffusion models mentioned here have been developed by others and have been shown above to be representable via GAPs, none of these papers has developed algorithms to solve SNDOP queries. We emphasize that not only do we give algorithms to answer SNDOP queries in the next section, our algorithms take any arbitrary diffusion model that can be expressed as a GAP, and an objective function as input. In addition, our notion of a social network is much more general than that of many existing approaches.

5. ALGORITHMS

In this section we study how to solve SNDOPs algorithmically.

5.1. Naive Algorithm

The naive algorithm for solving a SNDOP query is to first find all pre-answers to the query. Then compute the function $value$ for each pre-answer and find the best. This is obviously an extremely expensive algorithm that is unlikely to terminate in a reasonable amount of time.

An execution strategy that first finds all vertices in a social network $\mathcal{S}$ that satisfy the vertex condition and then somehow restricts interest to those vertices in the above computation (where $\mathcal{S}$ is embedded in a GAP Π) would not be correct for two reasons. First, $lfp(\mathbf{T}_\Pi)$ assigns a truth value to each ground vertex atom A that might be different from what is initially assigned within the social network. Second, when we add a new ground vertex atom A to Π (e.g. in our cell phone example, when we consider the possibility of assigning a free calling plan to a vertex v), it might be the case that vertices that previously did not satisfy the vertex condition VC do so after the addition of A to Π.

5.2. A Non-Ground Algorithm in the Monotonic Case

There are three major problems with the Naive algorithm. The first problem is that the aggregate function is very general and has no properties that we can take advantage of. Hence, we can show that the entire search space might need to be explored if an arbitrary aggregate function is used. The second problem is that it works on the "ground" instantiation of Π. The third problem is that the $\mathbf{T}_\Pi$ operator maps all *ground* atoms to the $[0, 1]$ interval and there can be a very large number of ground atoms to consider. For instance, if we have a very small social network with just 1000 vertices and a rule with 3 variables in it, that rule has 10^9 possible ground instances - an enormous number. *All these problems are further aggravated by the fact that fixpoints might have to be computed several times*.

In this section, we provide an algorithm to compute answers to a SNDOP query *under the assumption* that our aggregate function is *monotonic* and under the assumption that all rules in a GAP have the form $A : f(\mu_1, \ldots, \mu_n) \leftarrow B_1 : \mu_1, \wedge \cdots, B_n : \mu_n$, where each μ_i is a member of $[0, 1] \cup \mathsf{AVar}$.

In this case, we define a *non-ground* interpretation and a *non-ground* fixpoint operator $\mathbf{S}_\Pi$. This leverages existing work on non-ground logic programming initially pioneered by [M. Falaschi and Palamidessi 1988] and later adapted to different logic programming extensions by [Gottlob et al. 1996; Eiter et al. 1997; Stroe and Subrahmanian 2003]. We will use $\mathcal{A}^*$ to denote the set of all atoms (ground and non-ground). We start by defining a non-ground interpretation.

Definition 5.1. A *non-ground interpretation* is a *partial* mapping $NG : \mathcal{A}^* \rightarrow [0, 1]$. Every non-ground interpretation NG *represents* an interpretation $grd(NG)$ defined as follows: $grd(NG)(A) = \mathsf{max}\{NG(A') \mid A \text{ is a ground instance of } A'\}$. When there is no

atom A' which has A as a ground instance and for which $NG(A')$ is defined, then we set $grd(NG)(A) = 0$.

Thus, in a language with just three constants a, b, c and one predicate symbol p, the non-ground interpretation that maps $p(X, a)$ to 0.5 and everything else to 0 corresponds to the interpretation that assigns 0.5 to $p(a, a), p(b, a)$ and $p(c, a)$ and 0 to every other ground atom. Non-ground interpretations are succinct representations of ordinary interpretations - they try to keep track only of assignments to non-ground atoms (not necessarily all ground atoms) and they do not need to worry about atoms assigned 0. We now define a fixpoint operator that maps non-ground interpretations to non-ground interpretations.

Definition 5.2 (*operator* $\mathbf{S}_\Pi$). The operator $\mathbf{S}_\Pi$ associated with a GAP Π maps a non-ground interpretation NG to the non-ground interpretation $\mathbf{S}_\Pi(NG)$ where $\mathbf{S}_\Pi(NG)(A') = \mathsf{max}\{f(X_1, \ldots, X_n) \mid A : f(\mu_1, \ldots, \mu_n) \leftarrow B_1 : \mu_1 \wedge \ldots \wedge B_n : \mu_n$ is a rule in Π and there exist atoms $(B'_1, \ldots, B'_n)$ such that $NG(B'_i)$ is defined for all $1 \leq i \leq n$, $(B_1, \ldots, B_n)$ and $(B'_1, \ldots, B'_n)$ are simultaneously unifiable via a most general unifier θ, $A' = A\theta$, and (i) if μ_i is a constant, then $NG(B'_i) \geq \mu_i$ – in this case $X_i = \mu_i$, and (ii) if μ_i is a variable, then $X_i = NG(B'_i)\}$. (In this definition, without loss of generality, we assume the variables occurring in rules in Π are mutually standardized apart and are also different from those in NG).

The fixpoint operator $\mathbf{S}_\Pi$ delays grounding to the maximal extent possible by (i) only looking at the rules in Π directly rather than ground instances of rules in Π (which is what $\mathbf{T}_\Pi$ does), and (ii) by trying to assign values to non-ground atoms rather than ground instances - unless there is no other way around it. The following example shows how $\mathbf{S}_\Pi$ works.

Example 5.3. For illustrative purposes, suppose we have the following GAP Π:

$$e(V, a) : 1$$
$$p(V) : 0.7$$
$$q(V') : X \leftarrow p(V') : X, e(V', a) : 0.5$$

Let us apply $\mathbf{S}_\Pi$ till we reach a fixed point. With the first application, we entail the (non-ground) annotated atoms $e(V, a) : 1, p(V) : 0.7$ (we use the first and second facts). With the next application, $q(V') : 0.7$ is entailed. In fact, notice that the atoms in the body of the third rule, namely $p(V'), e(V', a)$, can be unified with $e(V, a), p(V)$, for which the interpretation obtained in the first iteration is defined; moreover, the value of $e(V, a)$ in the interpretation is greater than 0.5. Notice also that in assigning a value to $q(V')$, the value of $p(V')$ in the current interpretation is used, that is, 0.7 is used in place of X. At this point the least fixed point is reached.

Consider the ordering $\preceq$ defined as follows on non-ground interpretations: $NG_1 \preceq NG_2$ iff $grd(NG_1) \leq grd(NG_2)$. In this case, it it easy to see that:

PROPOSITION 5.4. *Suppose* Π *is any GAP. Then:*

(*1*) $\boldsymbol{S}_\Pi$ *is monotonic.*
(*2*) $\boldsymbol{S}_\Pi$ *has a least fixpoint* $lfp(\boldsymbol{S}_\Pi)$ *and* $lfp(\mathbf{T}_\Pi) = grd(lfp(\boldsymbol{S}_\Pi))$. *That is,* $lfp(\boldsymbol{S}_\Pi)$ *is a non-ground representation of the (ground) least fixpoint operator* $\mathbf{T}_\Pi$.

In short, $\mathbf{S}_\Pi$ is a version of $\mathbf{T}_\Pi$ that tries to work in a non-ground manner as much as possible. We now present the SNDOP-Mon algorithm to compute answers to a SNDOP query $(agg, VC, k, g_I(V), g_O(V))$ when agg is monotonic. The SNDOP-Mon algorithm uses the following notation: $value(NG)$ is the same as $value(grd(NG))$ and NG satisfies a formula iff $grd(NG)$ satisfies it.

SNDOP-Mon$(\Pi, agg, VC, k, g_I(V), g_O(V))$

(1) The variable $Curr$ is a tuple consisting of a GAP and natural number. We initialize $Curr.Prog = \Pi$; $Curr.Count = 0$.
(2) $Todo$ is a set of tuples described in step 1. We initialize $Todo \equiv \{Curr\}$
(3) Initialize the real number $bestVal = 0$ and GAP $bestSOL = NIL$
(4) **while** $Todo \not\equiv \emptyset$ **do**
  (a) $Cand =$ first member of $Todo$
  (b) **if** $value(lfp(\mathbf{S}_{Cand.Prog})) \geq bestVal \wedge lfp(\mathbf{S}_{Cand.Prog}) \models VC$ then
    i. $bestVal = value(lfp(\mathbf{S}_{Cand.Prog}))$; $bestSOL = Cand$
  (c) **if** $Cand.Count < k$ **then**
    i. For each ground atom $g_I(v)$, s.t. $\nexists OtherCand \in Todo$ where $OtherCand.Prog \supseteq Cand.Prog$, $|OtherCand.Prog| \leq |Cand.Prog| + 1$, and $lfp(\mathbf{S}_{OtherCand.Prog}) \models g_I(v) : 1$, do the following:
      A. Create new tuple $NewCand$.
        Set $NewCand.Prog = Cand.Prog \cup \{g_I(v) : 1\}$.
        Set $New.Count = Cand.Count + 1$)
      B. Insert $NewCand$ into $Todo$
    ii. Sort the elements of $Element \in Todo$ in descending order of $value(Element.Prog)$, where the first element, $Top \in Todo$, has the greatest such value (i.e. there does not exist another element Top' s.t. $value(Top'.Prog) > value(Top.Prog)$)
  (d) $Todo = Todo - \{Cand\}$
(5) **if** $bestSOL \neq NIL$ then **return** $(bestSOL.Prog - \Pi)$ **else** return NIL.

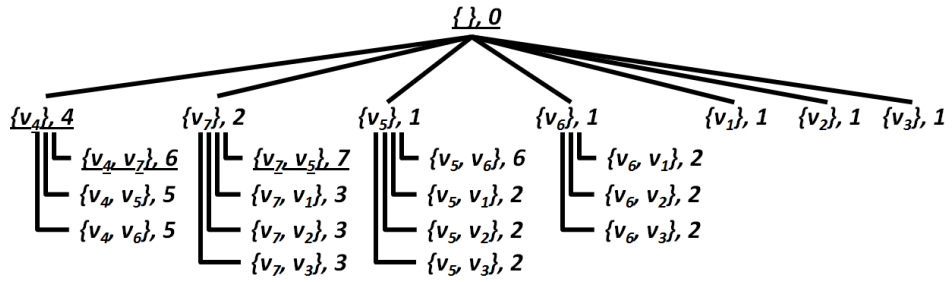


Fig. 5. Search tree for Example 5.5.

The following example shows how the SNDOP-Mon algorithm works.

Example 5.5. Consider Example 3.16 from page 14 where we present a social network and some diffusion rules for disease spread embedded in program $\Pi_{disease}$. Suppose, we want to answer a SNDOP query $(\Pi_{disease}, SUM, \textit{true}, 2, \textit{inf}(V), \textit{inf}(V))$. The search-tree in Figure 5 illustrates how SNDOP-Mon searches for an optimal solution to the query. In the figure, we labeled each node with the set of vertices and a real number. The vertices correspond to the vertex atoms (annotated with 1) formed with *inf* added to GAP in step 4(c)i. The real number corresponds to the value resulting from this addition. Underlined nodes in the search tree represent potential solutions where $bestVal$ and $bestSOL$ are updated. Notice, that, for example, the set $\{v_4, v_1\}$ is never considered. This is because $\textit{inf}(v_1)$ is entailed anytime a candidate solution includes v_4. The optimal solution is found to be $\{v_7, v_5\}$. In this example, the algorithm considers solutions in the following order: $\{\}, \{v_4\}, \{v_4, v_7\}, \{v_4, v_5\}, \{v_4, v_6\}, \{v_7\}, \{v_7, v_5\}, \{v_7, v_1\}, \{v_7, v_2\}, \{v_7, v_3\}, \{v_5\},$

$\{v_5, v_6\}, \{v_5, v_1\}, \{v_5, v_2\}, \{v_5, v_3\}, \{v_6\}, \{v_6, v_1\}, \{v_6, v_2\}, \{v_6, v_3\}, \{v_1\}, \{v_2\}, \{v_3\}$.

The following result states that the SNDOP-Mon algorithm is correct.

THEOREM 5.6. *Given SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$ and a GAP Π embedding a social network $\mathcal{S}$, if agg is monotonic then:*

- *There is an answer to the SNDOP query Q w.r.t. Π iff SNDOP-Mon$(\Pi, agg, VC, k, g_I(V), g_O(V))$ does not return NIL.*
- *If SNDOP-Mon$(\Pi, agg, VC, k, g_I(V), g_O(V))$ returns any result other than NIL, then that result is an answer to the SNDOP query Q w.r.t. Π.*

5.3. Approximation Algorithms: GREEDY-SNDOP

Even though SNDOP-Mon offers advantages such as pruning of the search tree and leverages non-ground operations to increase efficiency over the naive algorithm, it is still intractable in the worst case. Regretfully, Theorem 3.17 precludes an exact solution in PTIME and Theorem 3.20 precludes a PTIME α-approximation algorithm where $\alpha < \frac{e}{e-1}$. Both of these results hold for the restricted case of linear-GAPs and positive-linear aggregate functions.

The good news is that we were able to show that (i) for linear-GAPs and a-priori VC queries with positive-linear aggregates, the $value$ function is *submodular* (Theorem 3.15). (ii) Under these conditions, we can reduce the problem to the maximization of a submodular function over a uniform matroid (the uniformity of the matroid is proved in Lemma 3.14 for a-priori VC queries). (iii) We can leverage the work of [Nemhauser et al. 1978] that admits a greedy $\frac{e}{e-1}$ approximation algorithm. In this section, we develop a greedy algorithm for SNDOP queries that leverages the above three results.

The GREEDY-SNDOP algorithm shown below assumes a linear GAP, an a-priori VC query with positive-linear aggregates, and a zero-starting $value$ function (notice that the latter requirement can be met as stated by Proposition 3.11). The algorithm provides $\frac{e}{e-1}$ approximation to the SNDOP query problem. As this matches the upper bound of Theorem 3.20, we cannot do better in terms of an approximation guarantee.

GREEDY-SNDOP$(\Pi, agg, VC, k, g_I(V), g_O(V))$ returns $SOL \subseteq \mathbf{V}$

(1) Initialize $SOL = \emptyset$ and $\boldsymbol{REM} = \{v \in \mathbf{V} | \left(\{g_I(v) : 1\} \cup \bigcup_{pred \in \ell_{vert}(v)} \{pred(v) : 1\}\right) \models VC[V/v]\}$
(2) While $|SOL| < k$ and $\boldsymbol{REM} \neq \emptyset$
  (a) $v_{best} =$ null, $\boldsymbol{val} = value(SOL)$, $\boldsymbol{inc} = 0$
  (b) For each $v \in \boldsymbol{REM}$, do the following
    i. Let $\boldsymbol{inc}_{new} = value(SOL \cup \{v\}) - \boldsymbol{val}$
    ii. If $\boldsymbol{inc}_{new} \geq \boldsymbol{inc}$ then $\boldsymbol{inc} = \boldsymbol{inc}_{new}$ and $v_{best} = v$
  (c) $SOL = SOL \cup \{v_{best}\}$, $\boldsymbol{REM} = \boldsymbol{REM} - \{v_{best}\}$
(3) Return SOL

We now analyze the time complexity of GREEDY-SNDOP.

PROPOSITION 5.7. *Given a SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$, a social network $\mathcal{S}$, and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$, the complexity of GREEDY-SNDOP is $O(k \cdot |\mathbf{V}| \cdot F(|\mathbf{V}|))$ where $F(|\mathbf{V}|)$ is the time complexity to compute $value(\mathbf{V}')$ for some set $\mathbf{V}' \subseteq \mathbf{V}$ of size k.*

Table V. First iteration of the greedy algorithm

Vertex Atom	v_1	v_2	v_3	v_5	v_7	v_9	v_{10}
$buys\_camera(v_1)$	1.0	0.5	0.0	0.5	0.0	0.0	0.0
$buys\_camera(v_2)$	0.0	1.0	0.0	0.0	0.0	0.0	0.0
$buys\_camera(v_3)$	0.0	1.0	1.0	0.0	0.0	0.0	0.0
$buys\_camera(v_4)$	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$buys\_camera(v_5)$	0.0	0.0	0.0	1.0	0.0	0.0	0.0
$buys\_camera(v_6)$	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$buys\_camera(v_7)$	0.0	0.25	0.25	0.0	1.0	0.0	0.0
$buys\_camera(v_8)$	0.0	0.5	0.5	0.0	0.0	0.0	0.0
$buys\_camera(v_9)$	0.33	0.5	0.33	0.17	0.0	1.0	0.33
$buys\_camera(v_{10})$	0.0	0.5	0.5	0.0	0.0	0.0	1.0
SUM	1.33	4.25	2.58	1.67	1.0	1.0	1.33

Table VI. Incremental Increases for Both Iterations of GREEDY-SNDOP

Vertex	Incremental Increase on First Iteration	Incremental Increase on Second Iteration
v_1	1.33	0.67
v_2	4.25	NA
v_3	2.58	0.0
v_5	1.67	1.67
v_7	1.0	0.75
v_9	1.0	0.5
v_{10}	1.33	0.67

We note that most likely, the most expensive operation is the computation of $value$ at line 2(b)i. One obvious way to address this issue is by using a non-ground version of the fixed-point.

THEOREM 5.8. *Given a SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$, a social network $\mathcal{S}$, and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$, if*

— Π is a linear GAP,
— Q is a-priori VC,
— agg is positive-linear, and
— $value$ is zero-starting,

then GREEDY-SNDOP is an $(\frac{e}{e-1})$-approximation algorithm.

Example 5.9. Consider Example 4.1 and program Π_{lin} from page 19. Assume we have an additional vertex predicate symbol pro assigned to professional photographers (who are depicted with shaded vertices in Figure 3). Consider the SNDOP query where $agg =$ SUM, $VC = \{pro(V)\}$, $k = 2$, $g_I(V) = buys\_camera(V)$, $g_O(V) = buys\_camera(V)$.

On the first iteration of GREEDY-SNDOP, the algorithm computes the $value$ for all vertices in the set *REM* which are $v_1, v_2, v_3, v_5, v_7, v_9, v_{10}$. The resulting annotations of the fixed points and aggregates are shown in Table V.

As $value(\emptyset) = 0$, the incremental increase afforded by v_2 is 4.25 – and clearly the greatest of all the vertices considered. GREEDY-SNDOP adds v_2 to SOL, removes it from *REM* and proceeds to the next iteration. Table VI shows the incremental increases for the second iteration. As v_5 provides the greatest increase, it is picked, and the resulting solution is $\{v_2, v_5\}$.

6. IMPLEMENTATION AND EXPERIMENTS

We have implemented the GREEDY-SNDOP algorithm in 660 lines of Java code by re-using and extending the diffusion modeling Java library of [Broecheler et al. 2010]

(approx 35K lines of code). Our implementation uses multiple threads in the inner loop of the GREEDY-SNDOP algorithm to increase efficiency. All experiments were executed on the same machine with a dedicated 4-core 2.4GHz processor and 22GB of main memory. Times were measured to millisecond precision and are reported in seconds.

6.1. Experimental Setting

Data set. In order to evaluate GREEDY-SNDOP, we used a real-world dataset based on a social network of Wikipedia administrators and authors. Wikipedia is an online encyclopedia collaboratively edited by many contributors from all over the world. Selected contributors are given privileged administrative access rights to help maintain and control the collection of articles with additional technical features. A vote by existing administrators and ordinary authors determines whether an individual is granted administrative privileges. These votes are publicly recorded. [Leskovec et al. 2010] crawled 2794 elections from the inception of Wikipedia until January 2008. The votes casted in these elections give rise to a social network among Wikipedia administrators and authors by representing a vote of user i for user j as a directed edge from node i to j. In total, the dataset contains $103,663$ votes (edges) connecting more than 7000 Wikipedia users (vertices). Hence, the network is large and densely connected.[11]

SNDOP-Query. In our experiments, we consider the hypothetical problem of finding a set of administrators having the highest overall influence in the Wikipedia social network described above. We treat votes as a proxy for the inverse of influence. In other words, if user i voted for user j, we assume user j (intentionally through lobbying or unintentionally through the force of his contributions to Wikipedia) influenced user i to vote for him. All edges are assigned a weight of 1. Our SNDOP queries are designed as per the following definition.

Definition 6.1 (*Wikipedia SNDOP-Query*). Given some natural number $k > 1$, a Wikipedia SNDOP query, $WQ(k)$ is specified as follows:

— $agg =$ SUM – the intuition is that the aggregate provides us an expected number of vertices that are influenced.
— $VC = \emptyset$ – we do not use a vertex condition in our experiments
— k as specified by the input
— $g_I(V) = g_O(V) = influenced(V)$

Diffusion Models Used. We represented the diffusion process with two different models: one tipping and one cascading.

— **Cascading diffusion model.** We used the Flickr Diffusion Model (Diffusion Model 4.5 on page 22) described in Section 4.2. In this model, a constant parameter α represents the "strength" or "likelihood" of influence. The larger the parameter α the higher the influence of a user on those who voted for her.

— **Tipping diffusion model.** [Cha et al. 2009] shows that there is a relationship between the likelihood of a vertex marking a photo as a favorite and the percentage of their neighbors that also marked that photo as a favorite. This implies a tipping-model (as in Section 4.1). We apply the Jackson-Yariv model (i.e. Diffusion Model 4.2)

[11] Our Wikipedia data set does not include edge weights. However, including edge weights should not appreciably change the experimental results which show that solving SNDOP queries when tipping models are used is faster, in general, than when cascade models are used.

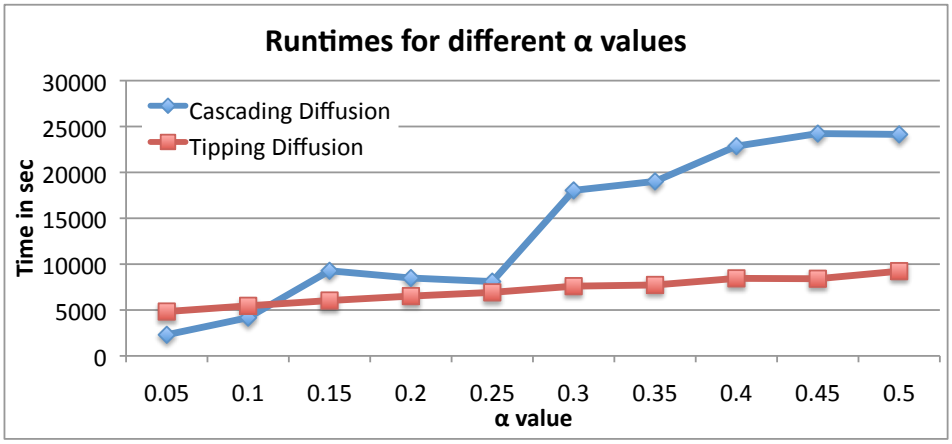


Fig. 6. Runtimes of GREEDY-SNDOP for different values of α and $k = 5$ in both diffusion models

with B equated to $influenced$. For each vertex $v_j \in \mathbf{V}$, we set the benefit to cost ratio ($\frac{b_j}{c_j}$) to 1. Finally, the function γ defined in the Jackson Yariv model is the constant-valued function (for all values of x):

$$\gamma(x) = \alpha.$$

This says that irrespective of the number of neighbors that a vertex has, the benefit to adopting strategy B (i.e. $influenced$) is α. Therefore, the resulting diffusion rule for the linear Jackson-Yariv model is:

$$influenced(v) : \alpha \cdot \frac{\sum_j X_j}{|\{v_j | \langle v_j, v \rangle\}|} \leftarrow \bigwedge_{v_j | \langle v_j, v \rangle \in \mathbf{E}} influenced(v_j) : X_j$$

For both models, we derive a unique logic program for each setting of the parameter α. The parameter α depends on the application and can be learned from ground truth data. In our experiments, we varied α to avoid introducing bias.

6.2. Experimental Results

Run-time of GREEDY-SNDOP with varying α and different diffusion models. Figure 6 shows the total runtime of GREEDY-SNDOP in seconds to find the set of $k = 5$ most influential users in the Wikipedia voting network for different values of the strength of influence parameter α. We varied α from 0.05 (very low level of influence) to 0.5 (very high level of influence) for both the cascading and tipping diffusion model. We observe that higher values of α lead to higher runtimes as expected since the scope of influence of any individual in the network is larger. Furthermore, we observe that the runtimes for the tipping diffusion model increase more slowly with α compared to the cascading model.

Run-time of GREEDY-SNDOP with varying k. For the next set of experiments, we keep the strength of influence fixed to $\alpha = 0.2$ and varied k which governs the size of the set of influencers. Figure 7 reports the runtime of GREEDY-SNDOP for the query $WQ(k)$ with $k = 5, 10, 15, 20, 25$. For the cascading model, the runtime is approximately linear in k – a curve-fitting analysis using Excel showed a slight superlinear trend (even though the figure itself looks linear at first sight). Figure 8 shows the time taken to execute each of the 25 iterations of the outer loop for the

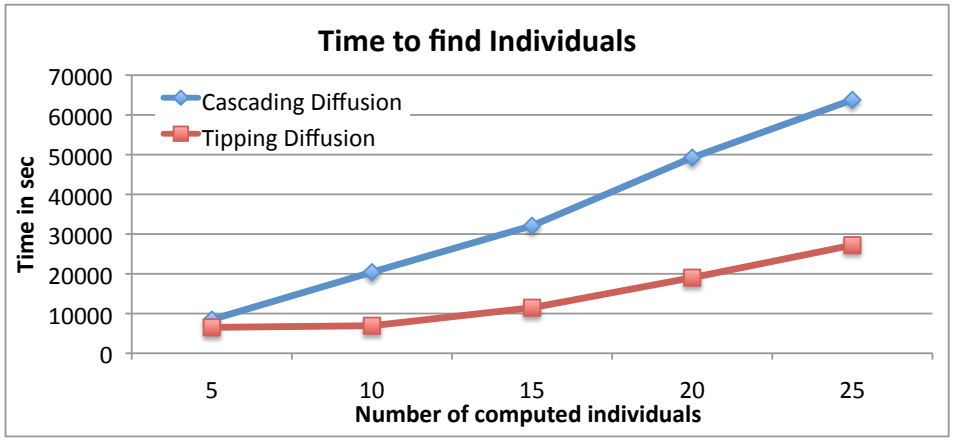


Fig. 7. Runtimes of GREEDY-SNDOP for different values of k and $\alpha = 0.2$ in both diffusion models

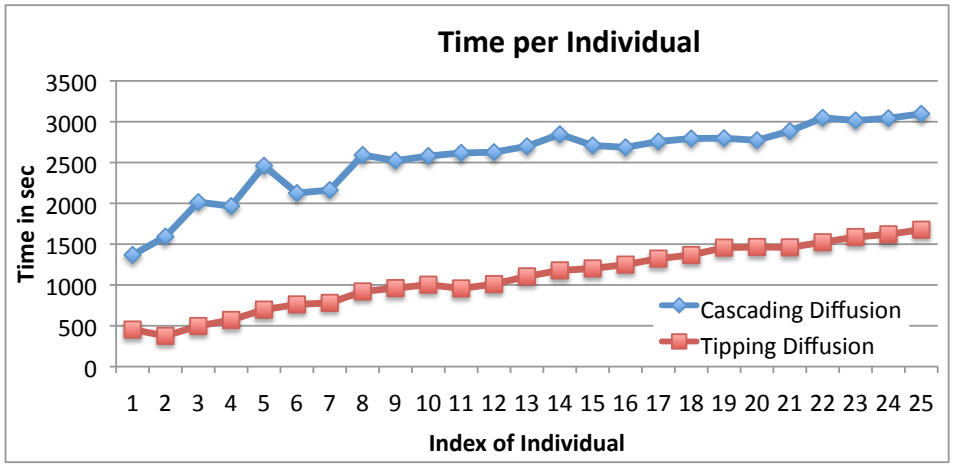


Fig. 8. Time per iteration of GREEDY-SNDOP for $\alpha = 0.2$ in both diffusion models

query $WQ(25)$ with $\alpha = 0.2$. Note that each subsequent iteration is more expensive than the previous one since the size of the logic programs to consider increases with the addition of each ground atom $influenced(v_i)$. However, we also implemented the practical improvement of "lazy evaluation" of the submodular function as described in [Leskovec et al. 2007b]. This improvement, which maintains correctness of the algorithms, stores previous improvements in total score and prunes the greedy search for the highest scoring vertex as discussed. We found that this technique also reduced the runtime of subsequent iterations.

Our experimental results show that we can answer SNDOP queries on large social networks. For example, computing the set of five most influential Wikipedia users in the voting network required approximately 2 hours averaged over the different values of α in the tipping diffusion model.

7. RELATED WORK

There has been extensive work in reasoning about diffusion in social networks. However, to our knowledge, there is no work on the relationship between logic programming and social networks. Moreover, there is no general framework to solve social network diffusion optimization problems that can take a broad class of diffusion models as input. We believe this work represents the first deterministic framework for representing generalized diffusion models that allows for different properties and weights on vertices and edges. Previously, the authors presented the framework of SNDOPs in [Shakarian et al. 2010]. However, this brief technical communication did not include either our exact or approximate algorithms, an implementation, experiments, the SNDOP-ALL problem, many of the complexity results, or many of the constructions seen in this paper (such as the homophilic diffusion models and big-seed marketing).

7.1. Related Work in Logic Programming

We first compare our work with annotated logic programming [Kifer and Subrahmanian 1992; Kifer and Lozinskii 1992; Thirunarayan and Kifer 1993] and its many extensions and variants [Vennekens et al. 2004; Krajci et al. 2004; Lu 1996; Lu et al. 1993; Damasio et al. 1999; Kern-Isberner and Lukasiewicz 2004; Lukasiewicz 1998]. There has been much work on annotated logic programming and we have built on the syntax and semantics of annotated LP. [Swift 1999] describes how lattice answer subsumption can implement GAPs whereas [Swift and Warren 2010] describes its implementation (as well as the implementation of partial order answer subsumption) in XSB and analyzes its performance showing scalability for applications in social network analysis, abstract interpretation, and query justification. We also note that possibilistic logic [Dubois and Prade 1990; Dubois et al. 1991] might be extended to handle the types of calculations GAPs support. In fact, GAPs may be viewed as such an extension of possibilistic logic. However, we are not aware of any work on solving optimization queries (queries that seek to optimize an aggregate function) w.r.t. annotated logic programming.

[Raedt et al. 2007] proposes a probabilistic version of Prolog, called *ProbLog*. A ProbLog program consists of a set of definite clauses (like in Prolog) where each clause is associated with a probability. Given a ProbLog program T, the authors induce a probability distribution over the space of definite logic programs $L \subseteq L_T$, where L_T is the definite logic program obtained from T by stripping out the probabilities. The probability of $L \subseteq L_T$ is obtained as $\prod_{c_i \in L} p_i \times \prod_{c_i \in L_T - L}(1 - p_i)$, where p_i is the probability of clause c_i. The probability that a query succeeds is the sum of the probabilities of the Prolog programs $L \subseteq L_T$ where the query succeeds. The semantics of ProbLog is called the *distribution semantics*; it has been borrowed from PRISM [Sato 1995]. Basically, in the distribution semantics all facts are assumed to be mutually independent [Hommersom and Lucas 2011]. Similar assumptions are made in certain other logics such as Independent Choice Logic [Poole 2008] and PRISM [Sato 1995; Sneyers et al. 2010]. Ng and Subrahmanian [Ng and Subrahmanian 1992; 1993] propose probabilistic logic programs where the independence assumption is not required - but this is computationally expensive though recent approaches [Khuller et al. 2007] based on sampling have been shown to scale very well to the case of 100K atoms. In order to compute the success probability of a query, [Raedt et al. 2007] first builds a monotone DNF formula (this represents the proofs of the query when probabilities are ignored), and then uses a BDD based approach to compute the probability. The approach is experimentally evaluated on biological networks showing good scalability.

The independence assumption is frequently made in many applications — however, in social networks, assuming independence of node properties and/or diffusions can be

dangerous because the diffusive process explicitly is one of dependency - the probability of vertex A being infected by a neighbor is directly dependent on the probabilities of one or more of its neighbors being infected. We note that [Raedt et al. 2007] does not provide any results on solving social network optimization problems.

In many logics that incorporate independence assumptions including [Raedt et al. 2007; Sneyers et al. 2010; Poole 2008], the probability of diffusion from neighbors of a vertex to a vertex are computed via the independence assumption. In the simplest sense, consider a vertex v and two vertices a, b such that $(a, v), (b, v)$ are edges in the graph and suppose there are no other edges of the form $(-, v)$. Suppose we know that the probability of v being infected by a (resp. b) is 0.7 (resp. 0.5). In this case, the probability of infection of v *under the assumption of independence* is $0.7+0.5-0.7\times0.5 = 0.85$. The reason independence is important here is that $\mathbf{P}(E \vee E') = \mathbf{P}(E) + \mathbf{P}(E') - \mathbf{P}(E \wedge E')$ and $\mathbf{P}(E \wedge E') = \mathbf{P}(E) \times \mathbf{P}(E')$ only when the events E, E' are mutually independent.

When independence is not assumed between E and E', one must compute $\mathbf{P}(E \wedge E')$ by either solving a linear program or via some other method. Dekhtyar et al. [Dekhtyar et al. 1999; Dekhtyar and Subrahmanian 2000] developed methods to not only find such probabilities when the independence assumption cannot be made, but also suggested how different assumptions on the relationships between events (e.g. positive correlation, negative correlation, mutual exclusion and independence) could be computed via hybrid logic programs. For example, if we assume that an arbitrary triangular co-norm[12] [Bonissone 1987b] $\oplus$ is used to compute the (disjunctive) probability that vertex v is infected by either a or by b, then we can express the diffusion via the GAP rule:

$$inf(v) : V_1 \oplus V_2 \leftarrow inf(a) : V_1 \wedge inf(b) : V_2.$$

Thus we see that such rules can capture triangular co-norms (including that used to compute the probability of a disjunct under the independence assumption).

However, though GAPs can be more expressive than many languages such as [Raedt et al. 2007; Sneyers et al. 2010; Poole 2008], there is no guarantee that they will be more "efficient" or more "intuitive" when additional assumptions such as independence are made. For instance, suppose we consider a more complex situation where we have three vertices a, b, c and the same vertex v above. And suppose we have edges $(a, v), (b, v), (c, v)$ in our graph and we want to say that infection propagation is independent. In this case, we can express this as a GAP by writing the rules shown below:

$$\begin{aligned}
inf(v) : V &\leftarrow edge(X, v) : 1 \wedge inf(X) : V. \\
inf(v) : V_1 \oplus V_2 &\leftarrow edge(X, v) : 1 \wedge edge(Y, v) : 1 \wedge \\
&\quad inf(X) : V_1 \wedge inf(Y) : V_2 \wedge X \neq Y. \\
inf(v) : V_1 \oplus V_2 \oplus V_3 &\leftarrow edge(X, v) : 1 \wedge edge(Y, v) : 1 \wedge edge(Z, v) : 1 \wedge \\
&\quad inf(X) : V_1 \wedge inf(Y) : V_2 \wedge inf(Z) : V_3 \wedge \\
&\quad X \neq Y \wedge Y \neq Z \wedge X \neq Z.
\end{aligned}$$

[12] A triangular co-norm is a function $\oplus : [0, 1] \times [0, 1] \rightarrow [0, 1]$ stating the probability of computing the "or" of two events whose probabilities are known and are provided as input to $\oplus$. All triangular co-norms satisfy the following axioms. **(Ax1)** $\oplus$ is associative and commutative. **(Ax2)** $x \oplus 0 = x$. **(Ax3)** $\oplus$ is monotone, i.e. if $x \leq x'$ and $y \leq y'$ then $x \oplus y \leq x' \oplus y'$. This axiom says that when the probabilities of both events go up (or stay the same), the probability of the "or" can only go up (or stay the same). Triangular co-norms have been extensively studied in logic programming as long back as 1988 [Subrahmanian 1988] — $max(x, y)$ and $x + y - x * y$ are two well known triangular co-norms — and there are many others.

When $x \oplus y = x + y - x * y$, then the above rules correspond to propagation under an independence assumption - otherwise $\oplus$ can be any triangular co-norm.

In this case, we wrote three rules, one for each possible number of infected predecessors of v. The first rule covers the case where we want the infection to pass from exactly one predecessor to v. The second rule covers all possible combinations of two predecessors of v passing the infection on. The third rule looks at the case where all three predecessors pass the infection along. Though this GAP is seemingly larger and more cumbersome than the simple conditional probability statement governing infections discussed above, we can show that indeed the third rule is the only one that is needed and this holds for any triangular co-norm — in fact, the semantics of the above GAP is identical to the semantics of the above GAP with just the last rule (plus the social network). Intuitively, the reason is that the probability of v getting the infection from all three is always greater than or equal to the probability of v getting it from just one or just two of its predecessors.

We now generalize the observations above. When we apply a triangular co-norm $\oplus$ to a set $\{x_1, \ldots, x_k\}$, we use $\oplus(\{x_1, \ldots, x_k\})$ as short-hand for $\oplus(x_1, \oplus(\{x_2, \ldots, x_n\}))$ which is well defined as all triangular co-norms are commutative and associative.

The following proposition says a triangular co-norm $\oplus$ applied to a set S' always gives a value no smaller than the value obtained by applying $\oplus$ to a subset of S'.

PROPOSITION 7.1. *Let S and S' be two sets of elements in $[0, 1]$, and $\oplus$ a triangular co-norm. If $S \subseteq S'$, then $\oplus S \leq \oplus S'$.*

PROOF. Suppose $S \subseteq S'$. We show that $\oplus S \leq \oplus(S \cup \Delta S)$ for any $\Delta S \subseteq S' - S$. This is shown by induction on the cardinality i of ΔS.
Base case $i = 0$. Straightforward.
Inductive step. Suppose $\oplus S \leq \oplus(S \cup \Delta S_i)$ for any $\Delta S_i \subseteq S' - S$ such that $|\Delta S_i| = i$. Let $\Delta S_{i+1} \subseteq S' - S$ be such that $|\Delta S_{i+1}| = i + 1$. Consider a set ΔS_i s.t. $\Delta S_i \subseteq \Delta S_{i+1}$ and $|\Delta S_i| = i$ and let e_i be the element in ΔS_{i+1} but not in ΔS_i. **Ax2** implies that $\oplus(S \cup \Delta S_i) = (\oplus(S \cup \Delta S_i)) \oplus 0$. By the associative and commutative properties of $\oplus$ we have $\oplus(S \cup \Delta S_{i+1}) = (\oplus(S \cup \Delta S_i)) \oplus e_i$. As $0 \leq e_i$, then $\oplus(S \cup \Delta S_i) \leq \oplus(S \cup \Delta S_{i+1})$ by the monotonicity property. As $\oplus S \leq \oplus(S \cup \Delta S_i)$ (by induction hypothesis), then $\oplus S \leq \oplus(S \cup \Delta S_{i+1})$. □

Thus, in order to deal with the infection scenario discussed above it suffices to write rules of the form:

$$\begin{aligned} inf(V) : V_1 \oplus \cdots \oplus V_n \leftarrow\ & edge(X_1, V) : 1 \wedge \cdots \wedge edge(X_n, V) : 1 \wedge \\ & inf(X_1) : V_1 \wedge \ldots \wedge inf(X_n) : V_n \wedge \\ & \bigwedge_{1 \leq i < j \leq n} X_i \neq X_j. \end{aligned}$$

Note that one such rule must be generated for each value n s.t. there is a node in the social network with in-degree n.

Thus, though GAPs provide a general method to express a vast variety of both probabilistic and non-probabilistic diffusion models, some of these methods can lead to an increase in the size of the GAP. When certain probabilistic assumptions are warranted (such as independence) then it may be appropriate to use the techniques for solving social network optimization problems presented in this paper in conjunction with frameworks such as ProbLog or Independent Choice Logic that may be able to leverage their assumptions to produce good solutions under those assumptions. However, much more

experimentation is required to understand the pros and cons of such choices and we leave this to future work.

There are a few papers on solving optimization problems in logic programming. The best of these is constraint logic programming [Van Hentenryck 2009] which can embed numerical computations within a logic program. However, CLP does not try to find solutions to optimization problems involving semantics structures of the program itself. Important examples of constraint logic programming include [Frühwirth 1994; Mancarella et al. 1999] where annotated LP is used for temporal reasoning, [Leone et al. 2004] assumes the existence of a cost function on models. They present an analysis of the complexity and algorithms to compute an optimal (w.r.t. the cost function) model of a disjunctive logic program in 3 cases: when all models of the disjunctive logic program are considered, when only minimal models of the disjunctive logic program are considered, and when stable models of the disjunctive logic program are considered. In contrast, in this paper, there are two differences. First, we are considering GAPs. Second, we are not looking for models of a GAP that optimize an objective function - rather, we are trying to find models of a GAP *together with some additional information* (namely some vertices in the social network for which a goal atom $g(v) : 1$ is added to the GAP) which is constrained (at most k additional atoms) so that the resulting least fixpoint has an optimal value w.r.t. an arbitrary $value$ function. In this regard, it has some connections with abduction in logic programs [Eiter and Gottlob 1995], but there is no work on abduction in annotated logic programs that we are aware of or work that optimizes an arbitrary objective function.

Our paper builds on many techniques in logic programming. It builds upon non-ground fixpoint computation algorithms proposed by [M. Falaschi and Palamidessi 1988] and later extended for stable models semantics [Gottlob et al. 1996; Eiter et al. 1997], and extends these non-ground fixpoint algorithms to GAPs and then applies the result to define the SNDOP-Mon algorithm to find answers to SNDOP queries which, to the best of our knowledge, have not been considered before.

7.2. Work in Social Networks

[Kempe et al. 2003] is one of the classic works in this area where a generalized diffusion framework for social networks is proposed. This work presents two basic diffusion models: the linear threshold and independent cascade models. Both models utilize random variables to specify how the diffusion propagates. These models roughly resemble non-deterministic versions of the tipping and cascading models presented in Section 4 of this paper. Neither model allows for a straightforward representation of multiple vertex or edge labels as this work does. Additionally, unlike this paper, where we use a fixed-point operator to calculate how the diffusion process unfolds, the diffusion models of [Kempe et al. 2003] utilize random variables to define the diffusion process and compute the expected number of vertices that have a given property. The authors of [Kempe et al. 2003] only approximate this expected value and leave the exact computation of it as an open question. Further, they provide no evidence that their approximation has theoretical guarantees. Moreover, Lemma 7.1 and the discussion immediately preceding it show how the linear threshold model mentioned in Kleinberg[Kempe et al. 2003] can be expressed via GAPs with no loss of generality (but with an increase in the number of rules in the GAP that can affect performance).

The more recent work of [Chen et al. 2010] showed this computation to be #P-hard by a reduction from S-T connectivity, which has no known approximation algorithm. This suggests that a reasonable approximation of the diffusion process of [Kempe et al. 2003] may not be possible. This contrasts sharply with the fixed-point operator of [Kifer and Subrahmanian 1992], which can be solved in PTIME under reasonable assumptions (which are present in this paper). [Kempe et al. 2003] focus on the problem of

finding the "most influential nodes" in the graph – which is similar in intuition to a SNDOP query. However, this problem only looks to maximize the expected number of vertices with a given property, not a complex aggregate as a SNDOP query does. Further, the approximation guarantee presented for the "most influential node" problem is contingent on an approximation of the expected number of vertices with a certain property, which is not shown (and, as stated earlier, was shown by [Chen et al. 2010] to be a #P-hard problem).

In short, the frameworks of [Chen et al. 2010] and [Kempe et al. 2003] cannot handle arbitrary aggregates nor vertex conditions nor edge and vertex predicates nor edge weights as we do. Nor can they define an objective function using a mix of the aggregate and the $g_O(-)$ predicate specified in the definition of a SNDOP query.

Another well-studied related problem in computer science is the "target set selection" problem [Dreyer and Roberts 2009; Chen 2009; Chiang et al. 2011]. This problem assumes a deterministic tipping model and seeks to find a set of vertices of a certain size that optimizes the final number of adopters. Although approximation algorithms for this problem have been discovered, there is no evidence that they scale well for large datasets. Further, an easy modification of Diffusion Model 4.1 allows for this problem to be represented in our framework. While target set selection can be encoded as an SNDOP query, a straightforward encoding of an SNDOP query into target set selection is unlikely. This is because the target set selection problem does not consider multiple vertex and edge labels nor seeks to optimize a complex aggregate.

8. CONCLUSION

Social networks are proliferating rapidly and have led to a wave of research on diffusion of phenomena in social networks. In this paper, we introduce the class of *Social Network Diffusion Optimization Problems* (SNDOPs for short) which try to find a set of vertices (where each vertex satisfies some user specified vertex condition) that has cardinality k or less (for a user-specified $k > 0$) and that optimizes an objective function specified by the user in accordance with a diffusion model represented via the well-known Generalized Annotated Program (GAP) framework. We have used specific examples of SNDOP queries drawn from product adoption (cell phone example) and epidemiology.

The major contributions of this paper include the following:

— We showed that answering SNDOP queries is NP-hard and identified the complexity classes associated with related problems (under various restrictions). We showed that the complexity of counting the number of solutions to SNDOP queries is #P-complete.
— We proved important results showing that there is no polynomial-time algorithm that computes an α-approximation to a SNDOP query when $\alpha \geq \frac{e}{e-1}$.
— We described how various well-known classes of diffusion models (cascading, tipping, homophilic) from economics, product adoption and marketing, and epidemiology can be embedded into GAPs.
— We presented an exact-algorithm for solving SNDOP queries under the assumption of a monotonic aggregate function.
— We proved that SNDOP queries are guaranteed to be submodular when the GAP representing the diffusion model is linear and the aggregate is positive-linear. We were able to leverage this result to develop the GREEDY-SNDOP algorithm that runs in polynomial-time and that achieves the best possible approximation ratio of $\frac{e}{e-1}$ for solving SNDOPs.
— We develop the first implementation for solving SNDOP queries and showed it could scale to a social network with over 7000 vertices and over 103,000 edges. Our exper-

iments also show that SNDOP queries over tipping models can generally be solved more quickly than SNDOP queries over cascading models.

Much work remains to be done and this paper merely represents a first step towards the solution of SNDOP queries. Clearly, we would like to scale SNDOP queries to social networks consisting of millions of vertices and billions of edges. This will require some major advances and represents a big challenge.

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

ACKNOWLEDGMENTS

Paulo Shakarian is funded under the US Army ACS/West Point Instructor (EECS) program.
Some of the authors of this paper were funded in part by AFOSR grant FA95500610405, ARO grant W911NF0910206 and ONR grant N000140910685.

REFERENCES

ANDERSON, R. M. AND MAY, R. M. 1979. Population biology of infectious diseases: Part i. *Nature 280,* 5721, 361.

APT, K. 2003. *Principles of constraint programming*. Cambridge University Press.

ARAL, S., MUCHNIK, L., AND SUNDARARAJAN, A. 2009. Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks. *Proceedings of the National Academy of Sciences 106,* 51, 21544–21549.

BACKSTROM, L., HUTTENLOCHER, D. P., KLEINBERG, J. M., AND LAN, X. 2006. Group formation in large social networks: membership, growth, and evolution. In *KDD*. 44–54.

BONISSONE, P. 1987a. Summarizing and propagating uncertain information with triangular norms. *International Journal of Approximate Reasoning 1,* 1, 71–101.

BONISSONE, P. P. 1987b. Summarizing and propagating uncertain information with triangular norms. *Int. J. Approx. Reasoning 1,* 1, 71–101.

BORGATTI, S. AND EVERETT, M. 2006. A graph-theoretic perspective on centrality. *Social Networks 28,* 4, 466–484.

BROECHELER, M., SHAKARIAN, P., AND SUBRAHMANIAN, V. 2010. A scalable framework for modeling competitive diffusion in social networks. *Social Computing / IEEE International Conference on Privacy, Security, Risk and Trust, 2010 IEEE International Conference on 0*, 295–302.

CENTOLA, D. 2010. The Spread of Behavior in an Online Social Network Experiment. *Science 329,* 5996, 1194–1197.

CENTOLA, D. 2011. An experimental study of homophily in the adoption of health behavior. *Science 334,* 6060, 1269–72.

CHA, M., MISLOVE, A., ADAMS, B., AND GUMMADI, K. P. 2008. Characterizing social cascades in flickr. In *WOSP '08: Proceedings of the first workshop on Online social networks*. ACM, New York, NY, USA, 13–18.

CHA, M., MISLOVE, A., AND GUMMADI, K. P. 2009. A Measurement-driven Analysis of Information Propagation in the Flickr Social Network. In *In Proceedings of the 18th International World Wide Web Conference (WWW'09)*. Madrid, Spain.

CHEN, N. 2009. On the approximability of influence in social networks. *SIAM J. Discret. Math. 23*, 1400–1415.

CHEN, W., WANG, C., AND WANG, Y. 2010. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '10. ACM, New York, NY, USA, 1029–1038.

CHIANG, C.-Y., HUANG, L.-H., LI, B.-J., WU, J., AND YEH, H.-G. 2011. Some results on the target set selection problem. *CoRR abs/1111.6685*.

COELHO, F., CODECO, C., AND CRUZ, H. 2008. Epigrass: A tool to study disease spread in complex networks. *Source Code for Biology and Medicin 3,* 3.

COWAN, R. AND JONARD, N. 2004. Network structure and the diffusion of knowledge. *Journal of Economic Dynamics and Control 28,* 8, 1557 – 1575.

DAMASIO, C., PEREIRA, L., AND SWIFT, T. 1999. Coherent well-founded annotated logic programs. In *Proc. Intl. Conf. on Logic Programming and Non-Monotonic Reasoning*. Springer Lecture Notes in Computer Science Vol. 1730, 262–276.

DEKHTYAR, A. AND SUBRAHMANIAN, V. S. 2000. Hybrid probabilistic programs. *J. Log. Program. 43,* 3, 187–250.

DEKHTYAR, M. I., DEKHTYAR, A., AND SUBRAHMANIAN, V. S. 1999. Hybrid probabilistic programs: Algorithms and complexity. In *UAI*. 160–169.

DREYER, P. AND ROBERTS, F. 2009. Irreversible -threshold processes: Graph-theoretical threshold models of the spread of disease and of opinion. *Discrete Applied Mathematics 157,* 7, 1615 – 1627.

DUBOIS, D., LANG, J., AND PRADE, H. 1991. A brief overview of possibilistic logic. In *ECSQARU*. 53–57.

DUBOIS, D. AND PRADE, H. 1990. Resolution principles in possibilistic logic. *Int. J. Approx. Reasoning 4,* 1, 1–21.

EITER, T. AND GOTTLOB, G. 1995. The complexity of logic-based abduction. *Journal of the ACM 42,* 1, 3–42.

EITER, T., LU, J., AND SUBRAHMANIAN, V. 1997. Computing non-ground representations of stable models. In *Proc. Intl. Conf. on Logic Programming and Non-Monotonic Reasoning*. Springer Lecture Notes in Computer Science Vol. 1265, 198–217.

FEIGE, U. 1998. A threshold of ln n for approximating set cover. *J. ACM 45,* 4, 634–652.

FRÜHWIRTH, T. 1994. Annotated constraint logic programming applied to temporal reasoning. In *PLILP: Programming Language Implementation and Logic Programming*. Springer, Madrid.

GAREY, M. R. AND JOHNSON, D. S. 1979. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.

GOTTLOB, G., MARCUS, S., NERODE, A., SALZER, G., AND SUBRAHMANIAN, V. S. 1996. A non-ground realization of the stable and well-founded semantics. *Theor. Comput. Sci. 166,* 1-2, 221–262.

GRANOVETTER, M. 1978. Threshold models of collective behavior. *The American Journal of Sociology 83,* 6, 1420–1443.

HETHCOTE, H. W. 1976. Qualitative analyses of communicable disease models. *Mathematical Biosciences 28,* 3-4, 335 – 356.

HOMMERSOM, A. AND LUCAS, P. J. F. 2011. Generalising the interaction rules in probabilistic logic. In *IJCAI*. 912–917.

JACKSON, M. AND YARIV, L. 2005. Diffusion on social networks. In *Economie Publique*. Vol. 16. 69–82.

KEMPE, D., KLEINBERG, J., AND TARDOS, E. 2003. Maximizing the spread of influence through a social network. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, New York, NY, USA, 137–146.

KERN-ISBERNER, G. AND LUKASIEWICZ, T. 2004. Combining probabilistic logic programming with the power of maximum entropy. *Artificial Intelligence 157,* 1-2, 139–202.

KHULLER, S., MARTINEZ, M. V., NAU, D. S., SLIVA, A., SIMARI, G. I., AND SUBRAHMANIAN, V. S. 2007. Computing most probable worlds of action probabilistic logic programs: scalable estimation for $10^{30,\,000}$ worlds. *Ann. Math. Artif. Intell. 51,* 2-4, 295–331.

KIFER, M. AND LOZINSKII, E. L. 1992. A logic for reasoning with inconsistency. *J. Autom. Reasoning 9,* 2, 179–215.

KIFER, M. AND SUBRAHMANIAN, V. 1992. Theory of generalized annotated logic programming and its applications. *J. Log. Program. 12,* 3&4, 335–367.

KLEINBERG, J. 2008. The convergence of social and technological networks. *Communications of the ACM 51,* 11, 66–72.

KOZEN, D. 1991. *The Design and Analysis of Algorithms*. Springer-Verlag, New York.

KRAJCI, S., LENCSES, R., AND VOJTS, P. 2004. A comparison of fuzzy and annotated logic programming. *Fuzzy Sets and Systems 144,* 1, 173 – 192.

LEONE, N., SCARCELLO, F., AND SUBRAHMANIAN, V. 2004. Optimal models of disjunctive logic programs: Semantics, complexity, and computation. *IEEE Transactions on Knowledge and Data Engineering 16*, 487–503.

LESKOVEC, J., ADAMIC, L. A., AND HUBERMAN, B. A. 2007a. The dynamics of viral marketing. *ACM Transactions on the Web 1,* 1.

LESKOVEC, J., HUTTENLOCHER, D., AND KLEINBERG, J. 2010. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World wide web*. WWW '10. ACM, New York, NY, USA, 641–650.

LESKOVEC, J., KRAUSE, A., GUESTRIN, C., FALOUTSOS, C., VANBRIESEN, J., AND GLANCE, N. 2007b. Cost-effective outbreak detection in networks. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, New York, NY, USA, 420–429.

LLOYD, J. W. 1987. *Foundations of logic programming*. Springer-Verlag New York, Inc.

LU, J. 1996. Logic programs with signs and annotations. *Journal of Logic and Computation 6,* 6, 755–778.

LU, J., MURRAY, N., AND ROSENTHAL, E. 1993. Signed formulas and annotated logics. In *Multiple-Valued Logic, 1993., Proceedings of The Twenty-Third International Symposium on*. 48–53.

LUKASIEWICZ, T. 1998. Probabilistic logic programming. In *Proceedings of ECAI '98*. 388–392.

M. FALASCHI, G. LEVI, A. M. AND PALAMIDESSI, C. 1988. A new declarative semantics for logic languages. In *Proc. 5th Internat. Conf. Symp. on Logic Programming*. 9931005.

MANCARELLA, P., RAFFAETÀ, A., AND TURINI, F. 1999. Temporal annotated constraint logic programming with multiple theories. In *DEXA '99: Proceedings of the 10th International Workshop on Database & Expert Systems Applications*.

MOSSEL, E. AND ROCH, S. 2007. On the submodularity of influence in social networks. In *Proc. STOC 2007*. ACM, New York, NY, USA, 128–134.

N. EAGLE, A. P. AND LAZER, D. 2008. Mobile phone data for inferring social network structure. In *Proc. 2008 Intl. Conference on Social and Behavioral Computing*. Springer Verlag, 79–88.

NEMHAUSER, G. L., WOLSEY, L. A., AND FISHER, M. 1978. An analysis of approximations for maximizing submodular set functions – I. *Mathematical Programming 14,* 1, 265–294.

NG, R. T. AND SUBRAHMANIAN, V. S. 1992. Probabilistic logic programming. *Inf. Comput. 101,* 2, 150–201.

NG, R. T. AND SUBRAHMANIAN, V. S. 1993. A semantical framework for supporting subjective and conditional probabilities in deductive databases. *J. Autom. Reasoning 10,* 2, 191–235.

POOLE, D. 2008. The independent choice logic and beyond. In *Probabilistic Inductive Logic Programming*. 222–243.

RAEDT, L. D., KIMMIG, A., AND TOIVONEN, H. 2007. Problog: A probabilistic prolog and its application in link discovery. In *IJCAI*. 2462–2467.

ROTH, D. 1996. On the hardness of approximate reasoning. *Artificial Intelligence 82*, 273–302.

RYCHTÁŘ, J. AND STADLER, B. 2008. Evolutionary dynamics on small-world networks. *International Journal of Computational and Mathematical Sciences 2,* 1.

SATO, T. 1995. A statistical learning method for logic programs with distribution semantics. In *ICLP*. 715–729.

SCHELLING, T. C. 1978. *Micromotives and Macrobehavior*. W.W. Norton and Co.

SHAKARIAN, P., SUBRAHMANIAN, V., AND SAPINO, M. L. 2010. Using generalized annotated programs to solve social network optimization problems. In *Technical Communications of the 26th International Conference on Logic Programming*, M. Hermenegildo and T. Schaub, Eds. Leibniz International Proceedings in Informatics (LIPIcs) Series, vol. 7. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 182–191.

SNEYERS, J., MEERT, W., VENNEKENS, J., KAMEYA, Y., AND SATO, T. 2010. Chr(prism)-based probabilistic logic learning. *TPLP 10,* 4-6, 433–447.

STROE, B. AND SUBRAHMANIAN, V. S. 2003. First order heterogeneous agent computations. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*. ACM, New York, NY, USA, 217–224.

SUBRAHMANIAN, V. 1988. Generalized triangular norm and co-norm based semantics for quantitate rule set logic programming.

SUBRAHMANIAN, V. S. AND RECUPERO, D. R. 2008. AVA: Adjective-Verb-Adverb Combinations for Sentiment Analysis. *IEEE Intelligent Systems 23,* 4, 43.

SUN, E., ROSENN, I., MARLOW, C., AND LENTO, T. 2009. Gesundheit! modeling contagion through facebook news feed. In *Proceedings of the Third International Conference on Weblogs and Social Media*. AAAI Press, AAAI Press, San Jose, CA.

SWIFT, T. 1999. Tabling for non-monotonic programming. *Ann. Math. Artif. Intell. 25,* 3-4, 201–240.

SWIFT, T. AND WARREN, D. S. 2010. Tabling with answer subsumption: Implementation, applications and performance. In *JELIA*. 300–312.

THIRUNARAYAN, K. AND KIFER, M. 1993. A theory of nonmonotonic inheritance based on annotated logic. *Artificial Intelligence 60,* 1, 23–50.

VAN HENTENRYCK, P. 2009. Constraint logic programming. *The Knowledge Engineering Review 6,* 03, 151–194.

VENNEKENS, J., VERBAETEN, S., AND BRUYNOOGHE, M. 2004. Logic programs with annotated disjunctions. In *Proc. Intl. Conf. on Logic Programming*. Springer Lecture Notes in Computer Science Vol. 3132, 431–445.

WATTS, D. AND PERETTI, J. 2007. Viral marketing for the real world. *Harvard Business Review*.

WATTS, D. J. 1999. Networks, dynamics, and the small-world phenomenon. *The American Journal of Sociology 105,* 2, 493–527.

ZHANG, L., M. P. 2011. Two is a crowd: Optimal trend adoption in social networks. In *Proceedings of International Conference on Game Theory for Networks (GameNets)*.

Received Received TBD; revised revised TBD; accepted accepted TBD

Online Appendix to: Using Generalized Annotated Programs to Solve Social Network Diffusion Optimization Problems

PAULO SHAKARIAN, United States Military Academy West Point
MATTHIAS BROECHELER, University of Maryland
V.S. SUBRAHMANIAN, University of Maryland
CRISTIAN MOLINARO, Università della Calabria

A. PROOFS FOR SECTION 3

A.1. Proof of Proposition 3.9

If agg is a positive-linear aggregate, then it is a monotonic aggregate.

PROOF. Follows directly from Definitions 3.7-3.8. □

A.2. Proof of Proposition 3.11

Let $Q = (agg, VC, k, g_I(V), g_O(V))$ be a SNDOP query which is not zero-starting w.r.t. a social network $\mathcal{S}$ and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$, and where agg is positive-linear. Let $agg'(X) = agg(X) - value(\emptyset)$. Then, $Q' = (agg', VC, k, g_I(V), g_O(V))$ is a SNDOP query which is zero-starting w.r.t. $\mathcal{S}$ and Π, $\mathsf{ans}(Q) = \mathsf{ans}(Q')$, and agg' is positive-linear.

PROOF. The fact that Q' is zero-starting and agg' is positive-linear follows directly from the construction. It is easy to see that $\mathsf{pre\_ans}(Q) = \mathsf{pre\_ans}(Q')$ as the set of pre-answers depends on Π, $\mathcal{S}$, VC, k, $g_I(V)$, and $g_O(V)$, which do not change for the two queries. We will use $value'$ to refer to the value function for Q'.
$\mathsf{ans}(Q) \subseteq \mathsf{ans}(Q')$. Reasoning by contradiction, assume that there is an answer $\mathbf{V}_{ans} \in \mathsf{ans}(Q)$ s.t. $\mathbf{V}_{ans} \notin \mathsf{ans}(Q')$. $\mathbf{V}_{ans} \in \mathsf{pre\_ans}(Q)$, which entails $\mathbf{V}_{ans} \in \mathsf{pre\_ans}(Q')$. Since $\mathbf{V}_{ans}$ is a pre-answer to Q' but not an answer, then there exists $\mathbf{V}'_{ans} \in \mathsf{pre\_ans}(Q')$ s.t. $value'(\mathbf{V}'_{ans}) > value'(\mathbf{V}_{ans})$. Then $\mathbf{V}'_{ans} \in \mathsf{pre\_ans}(Q)$ and $value(\mathbf{V}'_{ans}) = value'(\mathbf{V}'_{ans}) + value(\emptyset) > value(\mathbf{V}_{ans}) = value'(\mathbf{V}_{ans}) + value(\emptyset)$, that is $\mathbf{V}_{ans}$ is not an answer to Q, which is a contradiction.
$\mathsf{ans}(Q) \supseteq \mathsf{ans}(Q')$. A reasoning analogous to the one above can be applied. □

A.3. Proof of Lemma 3.13

Given a SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$, a social network $\mathcal{S}$, and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$, if agg is monotonic (Definition 3.7), then $value$ (defined as per Q and Π) is monotonic.

PROOF. By the definition of $\mathbf{T}$, the annotation of any vertex atom montonically increases as we add more facts of the form $g_I(V) : 1 \leftarrow$ to the logic program. Hence, by the monotonicity of agg, the statement follows. □

A.4. Proof of Lemma 3.14

Given a SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$, a social network $\mathcal{S}$, and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$, if Q is a-priori VC w.r.t. $\mathcal{S}$ and Π, then the set of pre-answers is a uniform matroid.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

PROOF. Let $\mathbf{V}_{cond}$ be the set of veritces in $\mathbf{V}$ s.t. for each $v \in \mathbf{V}_{cond}$, $\{g_I(v) : 1\} \cup \bigcup_{pred \in \ell_{vert}(v)} \{pred(v) : 1\} \models VC[V/v]$.
CLAIM 1: For an a-priori VC SNDOP query, any subset of $\mathbf{V}_{cond}$ of cardinality $\leq k$ is a pre-answer.
Suppose, BWOC, some subset of $\mathbf{V}' \subseteq \mathbf{V}_{cond}$ of cardinality $\leq k$ is not a pre-answer. Obviously, all such subsets meet the cardinality requirement. Then, there must exist some $v' \in \mathbf{V}'$ s.t. $\{g_I(v') : 1\} \cup \bigcup_{pred \in \ell_{vert}(v')} \{pred(v') : 1\} \not\models VC[V/v']$. By Definition 3.12, this is a contradiction.

CLAIM 2: There is no subset $\mathbf{V}' \subseteq \mathbf{V}$ where $\mathbf{V}' \cap (\mathbf{V} - \mathbf{V}_{cond}) \not\equiv \emptyset$ that is a pre-answer.
Clearly, this would have an element that would not satisfy the a-priori VC, and hence, not be a pre-answer.

Proof of lemma: Any subset of size $\leq k$ of $\mathbf{V}_{cond}$ is a uniform matroid by definition. Also, from claims 1-2, we know that this family of sets also corresponds exactly with the set of pre-answers. Hence, the statement of the lemma follows. □

A.5. Proof of Theorem 3.15

Given a SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$, a social network $\mathcal{S}$, and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$, if the following criteria are met:

— Π is a linear GAP,
— Q is a-priori VC, and
— agg is positive-linear,

then $value$ (defined as per Q and Π) is **sub-modular**.

In other words, for $\mathbf{V}_{cond} \equiv \{v' | v' \in \mathbf{V}$ *and* $(\Pi_{\mathcal{S}} \cup \{g_I(v') : 1\} \models VC[V/v'])\}$, if $\mathbf{V}_1 \subseteq \mathbf{V}_2 \subseteq \mathbf{V}_{cond}$ and $v \in \mathbf{V}_{cond} - \mathbf{V}_2$, then the following holds:

$$value(\mathbf{V}_1 \cup \{v\}) - value(\mathbf{V}_1) \geq value(\mathbf{V}_2 \cup \{v\}) - value(\mathbf{V}_2)$$

PROOF. CLAIM 1: For some $\mathbf{V}'$, if $A_i : \mu_i \in \mathbf{T}_{\{\Pi \cup \{g_I(v'):1\leftarrow \mid v' \in \mathbf{V}'\}}$ s.t. there is no $\mu_i' > \mu_i$ where $A_i : \mu_i' \in \mathbf{T}_{\{\Pi \cup \{g_I(v'):1\leftarrow \mid v' \in \mathbf{V}'\}}$ then, there exists a polynomial of the following form:

$$f_i(X_1, \ldots, X_{|\mathbf{V}|}) = \lambda_1 \cdot X_1 + \ldots + \lambda_{|\mathbf{V}|} \cdot X_{|\mathbf{V}|} + \lambda_{|\mathbf{V}|+1}$$

s.t. if each X_i where $V_i \in \mathbf{V}'$ is set to 1 and each X_i where $V_i \notin \mathbf{V}'$ is set to 0, then $f_i(X_1, \ldots, X_{|\mathbf{V}|}) = \mu_i$.
(Proof of claim 1): Consider all of the rules in $\{\Pi \cup \{g_I(v') : 1 \leftarrow$. If $A_i : \mu_i \in \mathbf{T}_{\{\Pi \cup \{g_I(v'):1\leftarrow \mid v' \in \mathbf{V}'\}}$, then there must exist a rule that causes the annotation of A_i to equal μ_i. As the annotation in all rules is a linear function, we can easily re-write it in the above form, based on the presence of annotated atoms in the body formed with the goal predicate.

CLAIM 2: For some $\mathbf{V}'$, if $A_i : \mu_i \in \mathbf{T}_{\{\Pi \cup \{g_I(v'):1\leftarrow \mid v' \in \mathbf{V}'\}} \uparrow j$, s.t. there is no $\mu_i' > \mu_i$ where $A_i : \mu_i' \in \mathbf{T}\{\Pi \cup \{g_I(v') : 1 \leftarrow \mid v' \in \mathbf{V}'\} \uparrow j$ then, there exists a polynomial of the following form:

$$f_i(X_1, \ldots, X_{|\mathbf{V}|}) = \lambda_1 \cdot X_1 + \ldots + \lambda_{|\mathbf{V}|} \cdot X_{|\mathbf{V}|} + \lambda_{|\mathbf{V}|+1}$$

s.t. if each X_i where $V_i \in \mathbf{V}'$ is set to 1 and each X_i where $V_i \notin \mathbf{V}'$ is set to 0, then $f_i(X_1, \ldots, X_{|\mathbf{V}|}) = \mu_i$.
(Proof of claim 2): We will show that if the statement of the claim is true for the

$j-1$ application of $\mathbf{T}$, then it is true for application j. The proof of the claim relies on this subclaim along with claim 1. If the claim holds for application $j-1$, then for each annotated atom $A'_i : \mu'_i$, there is an associated polynomial as per the statement. Consider the rule that fires in the jth application of the operator that causes rule A_i to be annotated with μ_i. We can re-write this as a polynomial of the above form, simply by substituting the polynomial for each annotation associated with A'_i from the previous iteration. As all of the polynomials are being substituted into variable positions of a polynomial, the result is still a polynomial, which can easily be re-arranged to resemble that of the claim.

CLAIM 3: For some $\mathbf{V}'$, if $A_i : \mu_i \in lfp(\mathbf{T}_{\{\Pi \cup \{g_I(v'):1\leftarrow \mid v' \in \mathbf{V}'\}})$, s.t. there is no $\mu'_i > \mu_i$ where $A_i : \mu'_i \in lfp(\mathbf{T}\{\Pi \cup \{g_I(v') : 1 \leftarrow \mid v' \in \mathbf{V}'\})$ then, there exists a polynomial of the following form:

$$f_i(X_1, \ldots, X_{|\mathbf{V}|}) = \lambda_1 \cdot X_1 + \ldots + \lambda_{|\mathbf{V}|} \cdot X_{|\mathbf{V}|} + \lambda_{|\mathbf{V}|+1}$$

s.t. if each X_i where $V_i \in \mathbf{V}'$ is set to 1 and each X_i where $V_i \notin \mathbf{V}'$ is set to 0, then $f_i(X_1, \ldots, X_{|\mathbf{V}|}) = \mu_i$.
(Proof of claim 3): Follows directly from claims 1-2.

CLAIM 4: For some $\mathbf{V}_i \subseteq \mathbf{V}$, there exists a polynomial of the following form:

$$f_i(X_1, \ldots, X_{|\mathbf{V}|}) = \lambda_1 \cdot X_1 + \ldots + \lambda_{|\mathbf{V}|} \cdot X_{|\mathbf{V}|} + \lambda_{|\mathbf{V}|+1}$$

s.t. if each X_i where $V_i \in \mathbf{V}'$ is set to 1 and each X_i where $V_i \notin \mathbf{V}'$ is set to 0, then $f_i(X_1, \ldots, X_{|\mathbf{V}|}) = \textit{value}(\mathbf{V}_i)$.
(Proof of claim 4): Consider all atoms formed with predicate $goal$ in the lfp where the annotation is maximum. By claim 3, each is associated with a polynomial. A positive-linear combination of all these polynomials is a polynomial of the form in this claim, and is equivalent to *value*.

CLAIM 5: $value(\mathbf{V}_1 \cup \{v\}) - value(\mathbf{V}_1) \geq value(\mathbf{V}_2 \cup \{v\}) - value(\mathbf{V}_2)$.
(Proof of claim 5): By the definition of $value$, as the query is a-priori VC, we know that $value$ is defined on all subsets of $\mathbf{V}_{cond}$.
We define the following polynomial functions, which are associated with *value* for the various subsets of $\mathbf{V}$ in claim 5 (with some re-arrangement, Greek letters resemble constants, X variables can be either 0 or 1 - signifying if the associated subscript is includes in the associated set).

(1) $f_1(X_{\mathbf{V}_1}, X_{\{v\}}, X_{\mathbf{V}_2-\mathbf{V}_1}) = \alpha_1 \cdot X_{\mathbf{V}_1} + \beta_1 \cdot X_{\{v\}} + \gamma_1 \cdot X_{\mathbf{V}_2-\mathbf{V}_1} + \lambda_1$
$value(\mathbf{V}_1 \cup \{v\}) = f_1(1,1,0) = \alpha_1 + \beta_1 + \lambda_1$
(2) $f_2(X_{\mathbf{V}_1}, X_{\{v\}}, X_{\mathbf{V}_2-\mathbf{V}_1}) = \alpha_2 \cdot X_{\mathbf{V}_1} + \beta_2 \cdot X_{\{v\}} + \gamma_2 \cdot X_{\mathbf{V}_2-\mathbf{V}_1} + \lambda_2$
$value(\mathbf{V}_1) = f_2(1,0,0) = \alpha_2 + \lambda_2$
(3) $f_3(X_{\mathbf{V}_1}, X_{\{v\}}, X_{\mathbf{V}_2-\mathbf{V}_1}) = \alpha_3 \cdot X_{\mathbf{V}_1} + \beta_3 \cdot X_{\{v\}} + \gamma_3 \cdot X_{\mathbf{V}_2-\mathbf{V}_1} + \lambda_3$
$value(\mathbf{V}_2 \cup \{v\}) = f_3(1,1,1) = \alpha_3 + \beta_3 + \gamma_3 + \lambda_3$
(4) $f_4(X_{\mathbf{V}_1}, X_{\{v\}}, X_{\mathbf{V}_2-\mathbf{V}_1}) = \alpha_4 \cdot X_{\mathbf{V}_1} + \beta_4 \cdot X_{\{v\}} + \gamma_4 \cdot X_{\mathbf{V}_2-\mathbf{V}_1} + \lambda_4$
$value(\mathbf{V}_2) = f_4(1,0,1) = \alpha_4 + \gamma_4 + \lambda_4$

CLAIM 5.1: $\alpha_4 + \gamma_4 + \lambda_4 \geq \alpha_2 + \gamma_3 + \lambda_2$
(Proof of claim 5.1): We note that the constants in the f_i's defined earlier all correspond directly with constants seen in rules. Hence, as $f_4(1,0,1)$ corresponds with the maximum possible value for $value(\mathbf{V}_2)$, there can be no constants other than $\alpha_4, \gamma_4, \lambda_4$ that sum to a value greater than $value(\mathbf{V}_2)$. The statement of claim 5.1 immediately follows.

CLAIM 5.2: $\alpha_1 + \beta_1 + \lambda_1 \geq \alpha_3 + \beta_3 + \lambda_3$
(Proof of claim 5.2): Mirrors claim 5.1, (in this case, $value(\mathbf{V}_1 \cup \{v\})$ is the maximum possible value of $f_1(1,1,0)$).

(Completion of claim 5 / theorem): Suppose, BWOC, claim 5 is not true. Then, it must be the case that

$$value(\mathbf{V}_1 \cup \{v\}) - value(\mathbf{V}_1) < value(\mathbf{V}_2 \cup \{v\}) - value(\mathbf{V}_2)$$

This would imply:

$$\alpha_1 + \beta_1 + \lambda_1 + \alpha_4 + \gamma_4 + \lambda_4 < \alpha_3 + \beta_3 + \gamma_3 + \lambda_3 + \alpha_2 + \lambda_2$$

By claim 5.2, we have the following:

$$\alpha_4 + \gamma_4 + \lambda_4 < \gamma_3 + \alpha_2 + \lambda_2$$

Which contradicts claim 5.1. The statement of the theorem follows. □

A.6. Proof of Theorem 3.17

Finding an answer to a SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$ (w.r.t. a social network $\mathcal{S}$ and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$) is NP-hard (even if Π is a linear GAP, $VC = \emptyset$, $agg = SUM$ and $value$ is zero-starting).

PROOF. The known NP-hard problem of max k-cover [Feige 1998] as follows.
MAX K-COVER
INPUT: Set of elements, S and a family of subsets of S, $\mathcal{H} \equiv \{H_1, \ldots, H_{max}\}$, and positive integer K.
OUTPUT: $\leq K$ subsets from $\mathcal{H}$ s.t. the union of the subsets covers a maximal number of elements in S.

We shall make the following assumptions of **MAX-K-COVER**

(1) $|\mathcal{H}| > K$
(2) There is no $H \in \mathcal{H}$ s.t. $H \equiv \emptyset$

CONSTRUCTION: Given **MAX K-COVER** input $S, \mathcal{H}, K$ we create a SNDOP-query as follows.

(1) Set up social network $\mathcal{S}$ as follows:
  (a) $\mathsf{EP} \equiv \{edge\}$
  (b) $\mathsf{VP} \equiv \{vertex\}$
  (c) For every element of $\mathcal{H}$, and every element of S, we create an element of V. We shall denote subsets of V, V_S and $V_{\mathcal{H}}$ as the vertices corresponding with S and $\mathcal{H}$ respectively. For some $s \in S$, v_s is the corresponding vertex. For some $H \in \mathcal{H}$, v_H is the corresponding vertex. Note that set $V \equiv V_S \cup V_{\mathcal{H}}$
  (d) For each $H \in \mathcal{H}$, if $s \in H$ draw add edge (v_H, v_s) to set E
  (e) For each $v \in V$, $\ell_{vert}(v) = vertex$
  (f) For each $(v, v') \in E$, $\ell_{edge}(v, v') = edge$
  (g) For each $(v, v') \in E$, $w(v, v') = 1$
(2) Set up program Π as follows:
  (a) Embed $\mathcal{S}$ into Π.
  (b) Add diffusion rule $vertex(V) : X \leftarrow vertex(V') : X \wedge edge(V', V) : 1$ to Π
(3) Set up SNDOP-query Q as follows:
  (a) $agg = SUM$
  (b) $VC = \emptyset$
  (c) $k = K$ (the K from SET_COVER)

(d) $g = vertex$

Additionally, we will use the following notation:

(1) V' is a pre-answer to the constructed query
(2) $value(V')$ is the value of the constructed query for pre-answer V'
(3) V'_{ans} is an answer to the constructed query

CLAIM 1: The construction can be performed in PTIME.
Straightforward.

CALIM 2: Program Π is a linear GAP.
Follows directly form Definition 3.6.

CLAIM 3: An answer V'_{ans} to the SNDOP query cannot contain a vertex in $v_s \in V_S$ **and** a vertex in $v_H \in V_{\mathcal{H}}$ s.t. $s \in H$.
BWOC, an optimal solution could have an element v_s as described in the claim. By assumption 1, there are more than K elements in $V_{\mathcal{H}}$ and all of them have an edge to some element of V_S by assumption 2. It is obvious that v_s will be annotated with a 1 in the fixed point, and that no elements of $V_{\mathcal{H}} - V'_{ans}$ will be annotated with 1 in the fixed point. Hence, we can pick any element of $V_{\mathcal{H}} - V'_{ans}$ and $value$ will be at least one greater than the "optimal" solution – hence a contradiction.

CLAIM 4: If an answer $V'_{ans} \cap V_S \not\equiv \emptyset$, then we can construct an alternative optimal solution such that $V'_{ans} \cap V_S \equiv \emptyset$.
As no element in $V'_{ans} \cap V_S \not\equiv \emptyset$ has an outgoing neighbor, and by assumption 1, we can be assured that $|V'_{ans} - V_{\mathcal{H}}| > |V'_{ans} \cap V_S|$, we can replace the elements of $V'_{ans} \cap V_S$ in V'_{ans} with elements from $V'_{ans} - V_{\mathcal{H}}$ and still be ensured of an optimal solution.

CLAIM 5: Given a set $\mathcal{H}' \subseteq \mathcal{H}$ that ensures an optimal solution to **MAX-K-COVER**, we can construct an optimal V'_{ans} to the SNDOP query.

CASE 1 (claim 5): $|\mathcal{H}'| = K$.
Let OPT be the number of elements of S covered in the optimal solution of **MAX-K-COVER**. For each $H \in \mathcal{H}'$, we pick the corresponding element of $V_{\mathcal{H}}$. Obviously, $value(V'_{ans}) = OPT + K$. Suppose, we could pick a different element of **V** and get a solution with a higher $value$. As no element of S has an outgoing edge, replacing one of the elements from the constructed set with one of these will not ensure a greater solution. If we could pick an element from $V_{\mathcal{H}} - V'_{ans}$, then this would obviously imply a solution to **MAX-K-COVER** s.t. more than OPT elements of S are covered – clearly this is a contradiction as $\mathcal{H}'$ is an optimal cover.

CASE 2 (claim 5): $|\mathcal{H}'| < K$.
Create $\mathcal{H}''$ with all of the elements of $\mathcal{H}'$ and $K - |\mathcal{H}'|$ elements of $\mathcal{H} - \mathcal{H}'$. Clearly, this is also an optimal solution to **MAX-K-COVER** (as cardinality is not optimized, just needs to be below K). We can now apply case 1 of this claim.

CLAIM 6: Given V'_{ans}, we can constructively create a subset of $\mathcal{H}$ that, if picked, ensures an optimal solution to **MAX-K-COVER**.

CASE 1 (claim 6): $V'_{ans} \subseteq V_{\mathcal{H}}$
Simply pick each H associated with each $v_H \in V'_{ans}$. Let $OPT' = value(V'_{ans})$ note that $OPT' = K + SPREAD$ where $SPREAD$ corresponds with the number of 1-annotated

elements of V_S in the fixed point. If there is a different subset of $\mathcal{H}$ that can be picked, (i.e. a more optimal solution to **MAX-K-COVER**), then we can create a solution to the SNDOP query where some $SPREAD' > SPREAD$ elements of V_S become annotated with 1 in the fixed point. Clearly, this would imply a more optimal solution to the SNDOP query – a contradiction.

CASE 2 (claim 6): $V_S - V'_{ans} \not\equiv \emptyset$
From this solution, we can use claim 4 to create an optimal solution s.t. case 1 applies.

The proof of the theorem follows directly from claims 5-6. □

A.7. Proof of Theorem 3.18

Given a SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$, a social network $\mathcal{S}$, a GAP $\Pi \supseteq \Pi_S$, and a real number $target$, the problem of checking whether there exists a pre-answer $\mathbf{V'}$ s.t. $value(\mathbf{V'}) \geq target$ is in NP under the assumptions that agg and the functions in $\mathcal{F}$ are polynomially computable, and Π is ground.

PROOF.
CLAIM 1: SNDOP-DEC is NP-hard.
We do this by a reduction from SET_COVER.

CONSTRUCTION: Given instance $S, \mathcal{H}, K$ of SET_COVER, we create K instances of SNDOP-DEC, each identified with index $i \in [1, K]$, that each use the same construction used to show the NP-hardness of a SNDOP query with the following two exceptions:

— Set k in SNDOP-DEC to i
— Set $target$ in SNDOP-DEC to $i + |S|$

CLAIM 1.1: The construction can be performed in PTIME.
Straightforward. CLAIM 1.2: If there is a solution to the set cover problem, at least one of the constructed instances of SNDOP-DEC will return "yes."
Suppose, that there is a solution to the set-cover problem, that causes the selection of m elements of $\mathcal{H}$ (where $m \leq K$). By the construction, there exists an instance of SNDOP-DEC such that $target = m + |S|$ and $k = m$. We simply pick the k vertices in $V_{\mathcal{H}}$ corresponding with the covers, and by the construction, after running Π, all of the vertices in V_S will have an annotation to the vertex atoms formed by *marked* of 1. Hence, the aggregate will be $m + |S|$ - which is greater than $target$, so that instance of SNDOP-DEC returns "yes."
CLAIM 1.3: If there is no solution to the set cover problem, all of the instances of SNDOP-DEC will return "no."
Suppose there is no solution to SET_COVER and one of the constructed instances of SNDOP-DEC returns "yes." Then, for some $i \in [1, K]$, there are i vertices that can be picked to change the annotation of the $vertex$ vertex atoms to ensure that the aggregate is greater than or equal to $i + |S|$. As, at most, only i vertex atoms can be picked, and only atoms in V_S can change annotation due to Π, all i vertices associated with the vertex atoms must be in $V_{\mathcal{H}}$ to ensure that we have the most possible vertex atoms formed with $vertex$ that have a non-zero annotation. However, in order for all of the vertices in V_S to have the annotations of the associated $vertex$ vertex atom increase to 1, there must be at least one incoming edge to each element of V_S from one of the i atoms from $V_{\mathcal{H}}$. By how $\mathcal{S}$ is constructed, this would imply a set-cover of size i, which would be a contradiction.
PROOF OF CLAIM 1: Follows directly from claims 1.1-1.3.

CLAIM 2: SNDOP-DEC is in-NP (with the conditions in the statement).
Suppose, we are given a set V'. We can easily verify this solution in PTIME as follows: (i) verify V' is a valid pre-answer can easily be done in PTIME by checking that $|V'| \leq k$ and that $\forall v' \in V'$, $VC(v')$ is true. (ii) by the assumptions about agg and the functions in $\mathcal{F}$, we can compute $value(V')$ in PTIME as well. the statement follows. □

A.8. Proof of Theorem 3.20

Answering a SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$ (w.r.t. a social network $\mathcal{S}$ and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$) cannot be approximated in PTIME within a ratio of $\frac{e-1}{e} + \epsilon$ for some $\epsilon > 0$ (where e is the inverse of the natural log) unless $\mathbf{P} = \mathbf{NP}$ – even if Π is a linear GAP, $VC = \emptyset$, $agg = SUM$ and $value$ is zero-starting.

PROOF. Suppose, BWOC, there is an α-approximation algorithm for an SNDOP query. Hence, we can approximate $value$ returned by SNDOP within a factor of $1 - 1/e + \epsilon$ for some $\epsilon > 0$. Using the **MAX-K-COVER** reduction in Theorem 3.17, for SNDOP answer V'_{ans}, the cardinality of the covered elements of S in **MAX-K-COVER** is $value(V'_{ans}) - K$. Hence, this approximation algorithm would provide a solution to **MAX-K-COVER** within a factor of $1 - 1/e + \epsilon$ for some $\epsilon > 0$. By Theorem 5.3 of [Feige 1998], this would imply **P==NP**, which contradicts the statement of the theorem. We recall that Theorem 5.3 of [Feige 1998] states that for any $\epsilon > 0$, **MAX-K-COVER** cannot be approximated in polynomial time within a ratio of $(1 - 1/e + \epsilon)$, unless **P==NP**. The proof in [Feige 1998] uses a reduction from approximating 3SAT-5, which is the problem of determining the maximum number of clauses that can be simultaneously satisfied in a CNF formula with n variables and $5n/3$ clauses, in which every clause contains exactly three literals, every variable appears in exactly five clauses, and a variable does not appear in a clause more than once. □

A.9. Proof of Theorem 3.21

Counting the number of answers to a SNDOP query Q (w.r.t. a social network $\mathcal{S}$ and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$) is #P-complete.

Follows directly from Lemmas A.1 and A.2.

LEMMA A.1. *The counting version of the SNDOP query answering problem (we shall call it #SNDOP) is #P-hard.*

PROOF. We now define the known #P-Complete problem, MONSAT [Roth 1996] and a variant of it used in this proof:
Counting K-Monotone CNF Sat. (#MONSAT)
INPUT: Set of clauses C, each with K disjuncted literals, no literals are negations, L is the set of atoms.
OUTPUT: Number of subsets of L such that if the atoms in the subset are true, all of the clauses in C are satisfied.

Counting K-Monotone CNF Sat. - Exact (#MONSAT-EQ)
INPUT: Set of clauses C, each with K disjuncted literals, no literals are negations, L is the set of atoms and natural number m.
OUTPUT: Number of subsets of L - each with cardinality of exactly m - such that if the atoms in the subset are true, all of the clauses in C are satisfied.

We now define the following problem used in the proof:
#SNDOP-EQ
INPUT: Same as SNDOP-DEC.
OUTPUT: Number of pre-answers V' that would causes a "yes" answer to SNDOP-

DEC *and* $|V'| = k$.

CLAIM 1: #MONSAT$\leq_p$#MONSAT-EQ and #MONSAT-EQ is #P-hard Consider the following construction (CONSTRUCTION 1):
Let L be the set of atoms associated with #MONSAT. Create $|L|$ instances of #MONSAT-EQ - each with a cardinality constraint (m) in $[1, |L|]$, and the remainder of the input the same as #MONSAT.
(Proof of claim 1): The sum of the solution to the $|L|$ instances of #MONSAT-EQ is equal to the solution to #MONSAT.
Every possible satisfying assignment counted as a solution to #MONSAT has a unique cardinality associated with it, which is in $[1, |L|]$. The claim follows trivially from this fact and construction 1 (which can be performed in PTIME).

CLAIM 2: #MONSAT-EQ$\leq_p$#SNDOP-EQ and #SNDOP-EQ is #P-hard
Consider the following construction (CONSTRUCTION 2):
Given #MONSAT-EQ input (C, K, L, m), we create an instance of #SNDOP-EQ as follows.

(1) Set up social network $\mathcal{S}$ as follows:
  (a) EP $\equiv \{edge\}$
  (b) VP $\equiv \{vertex\}$
  (c) For every element of C, and every element of L, we create an element of V. We shall denote subsets of V, V_C and V_L as the vertices corresponding with C and L respectively. For some $a \in C$, v_a is the corresponding vertex. For some $b \in L$, v_b is the corresponding vertex.
  (d) For each $a \in C$, if b is in clause C, add edge (v_b, v_a) to set E
  (e) For each $v \in V$, $\ell_{vert}(v) = vertex$
  (f) For each $(v, v') \in E$, $\ell_{edge}(v, v') = edge$
  (g) For each $(v, v') \in E$, $w(v, v') = 1$
(2) Set up program Π as follows:
  (a) Embed $\mathcal{S}$ into Π
  (b) For each $v \in V$, add fact $vertex(v) : 0$ to Π
  (c) Add diffusion rule $vertex(v) : 1 \leftarrow vertex(v') : 1 \wedge edge(v', v) : 1$ to Π
(3) Set up SNDOP-query Q as follows:
  (a) $agg = SUM$
  (b) $VC = \emptyset$
  (c) $k = m$ (the m from #MONSAT-EQ)
  (d) $g = vertex$
  (e) $target = |C| + k$

CLAIM 2.1: Construction 2 can be performed in PTIME.
Straightforward.

CLAIM 2.2: If there is a solution to given an instance of MONSAT-EQ, then given construction 2 as input, SNDOP-EQ will return "yes". For each $a \in L$ in the solution to MONSAT-EQ, change the annotation of $vertex(v_a)$ to 1 in Π_{facts}. There are $m = k$ such vertices. By the construction, this will cause the $|C|$ vertices of V_C to increase their annotation - resulting in an aggregate of $|C| + k$, causing SNDOP-EQ to return "yes".

CLAIM 2.3: If, given construction 2 as input, SNDOP-EQ returns "yes", then a solution to given an instance of MONSAT-EQ such that k is the cardinality of the solution.
We note that selecting any vertex in V' not in V_L will result in an $value(V') < |C| + k$, as fewer than $|C|$ nodes will have their annotation increase after running Π. The only

way to achieve an $value(V') = |C| + k$ is if there exists a set of k vertices in V_L such that there is an outgoing edge from at least one of the picked vertices to each node in V_C. This is only possible if there exists a solution to the MONSAT-EQ problem.

CLAIM 2.4: There is a 1-1 correspondence between solution to MONSAT-EQ and SNDOP-EQ using construction 2.
As each literal in a MONSAT-EQ solution corresponds to exactly one vertex in a SNDOP-EQ, and by claims 2.2-2.3, the claim follows.

PROOF OF CLAIM 2: Follows directly from claims 2.1-2.4.

CLAIM 3: #SNDOP-EQ$\leq_p$#SNDOP, #SNDOP is #P-hard
Consider the following construction (CONSTRUCTION 3):
Let k be the cardinality constraint associated with #SNDOP-EQ. Create two instances of #SNDOP, one with a cardinality constraint of k and one with the constraint of $k-1$, and the remainder of the input is the same as #SNDOP-EQ.
PROOF OF CLAIM 3: First, note that construction 3 can be performed in PTIME. We show that the solution to #SNDOP with cardinality constraint $k-1$ subtracted from the solution to #SNDOP with cardinality constraint k is the solution to #SNDOP-EQ. As the solution to #SNDOP with cardinality constraint $k-1$ is the number of all V''s that are a solution with cardinality of $k-1$ or less, and the solution to #SNDOP with cardinality constraint k is the number of all V''s that are a solution with cardinality of k or less, the difference is the number of all V''s with a cardinality of exactly k.

PROOF OF LEMMA: Follows directly from claims 3. □

LEMMA A.2. *If the aggregate function agg is polynomially computable and functions in $\mathcal{F}$ are polynomially computable, then #SNDOP is in-#P.*

PROOF. We use the two requirements for membership in-#P as presented in [Kozen 1991].
(i) Witnesses must be verifiable in PTIME (shown in the NP-Completness of a SNDOP-query).
(ii) The number of solutions to #SNDOP is bounded by $x'^{k'}$ - where k' is a constant. We know that the number of solutions is bounded by $\sum_{i\leq k}\binom{|V|}{i}$ which is less than $c\cdot|V|^k$ for some constant c. □

A.10. Proof of Theorem 3.22

Given a SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$, a social network $\mathcal{S}$, and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$), there exists a polynomial-time algorithm with an oracle to SNDOP-ALL which answers Q.

PROOF. We shall refer to the problem of finding $\bigcup_{V'_{ans}\in \mathsf{ans}(Q)} V'_{ans}$ as SNDOP-ALL. We show that SNDOP-ALL is $\leq_p$ solving a SNDOP-query.

Given set an instance of SNDOP-ALL and vertex set V^*, $|V^*| \leq k$ let SNDOP-ALL(V^*) be the modification of of the instance of SNDOP-ALL where the value k is reduced by $|V^*|$ and for each $v_j^* \in V^*$, the fact $g_I(v_i) : 1$ is added to Π.

Consider the following informal algorithm (FIND-SET) that takes an instance of SNDOP-ALL (Q) and some vertex set V^*, $|V^*| \leq k$.

(1) If $|V^*| = k$, return V^*
(2) Else, solve SNDOP-ALL(V^*), returning set V''.
    (a) If $V'' - V^* \equiv \emptyset$, return V^*

(b) Else, pick $v \in V'' - V^*$ and return FIND-SET($Q, V^* \cup v$)

Note, that the above algorithm can only iterate k times.
CLAIM 1: The V^* returned by FIND-SET is a valid solution to the SNDOP-query (with the same input for Q).
First, we number the elements in V^* as $v_1, \ldots, v_{size}$ - where v_1 is picked as the first element in the solution and vertex v_i is added at the ith recursive call of FIND-SET. We know that $size \leq k$
BASE CASE: There is a set of vertices of size $\leq size$ that is a solution to the SNDOP-query s.t. vertex v_1 is in that set - follows directly from the definition of SNDOP-ALL.
INDUCTIVE HYPOTHESIS: For some $k' \leq size$, we assume that for vertices $v_1, \ldots, v_{k'-1}$ there is some set of vertices of size $\leq k$ that is a solution to the SNDOP-query s.t. vertices $v_1, \ldots, v_{k'-1}$ are in that set.
INDUCTIVE STEP: For some $k' \leq size$, consider vertices $v_1, \ldots, v_{k'}$. By the inductive hypothesis, vertices $v_1, \ldots, v_{k'-1}$ are in a $\leq k$-sized solution. By the construction, and the definition of SNDOP-ALL, we know that vertex $v_{k'}$ must also be in that set as well.

CLAIM 2: Given some V' as a solution to the SNDOP-query, the algorithm FIND-SET can be run in such a way to return that set.
Number each vertex in V' as $v_1, \ldots, v_{size}$. By the definition of SNDOP-ALL, upon the i'th call to FIND-SET, we are guaranteed that the vertices $v_i, \ldots, v_{size}$ will be in set V''. Simply pick vertex v_i follow the algorithm to the next recursive call, the claim immediately follows.
PROOF OF PROPOSITION: Note the construction can be accomplished in PTIME. The proposition follows directly from claims 1-2. □

A.11. Proof of Theorem 3.23

Given a SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$, a social network $\mathcal{S}$, and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$), finding $\bigcup_{\mathbf{V}' \in \text{ans}(Q)} \mathbf{V}'$ reduces to $|\mathbf{V}| + 1$ SNDOP queries, where $\mathbf{V}$ is the set of vertices of $\mathcal{S}$.

PROOF. We set up $|V|$ SNDOP-queries as follows:

— Let k_{all} be the k value for the SNDOP-ALL query and and for each SNDOP-query i, let k_i be the k for that query. For each query i, set $k_i = k_{all} - 1$.
— Number each element of $v_i \in V$ such that $g_I(v_i)$ and $VC(v_i)$ are true. For the ith SNDOP-query, let v_i be the corresponding element of V
— Let Π_i refer to the GAP associated with the ith SNDOP-query and Π_{all} be the program for SNDOP-ALL. For each program Π_i, add fact $g_I(v_i) : 1$
— For each SNDOP-query i, the remainder of the input is the same as for SNDOP-ALL.

After the construction, do the following:

(1) We shall refer to a SNDOP-query that has the same input as SNDOP-ALL as the "primary query." Let ${V'_{ans}}^{(pri)}$ be an answer to this query and $value({V'_{ans}}^{(pri)})$ be the associated value.
(2) For each SNDOP-query i, let ${V'_{ans}}^{(i)}$ be an answer and $value({V'_{ans}}^{(i)})$ be the associated value.
(3) Let V'', the solution to SNDOP-ALL be initialized as $\emptyset$.
(4) For each SNDOP-query i, if $value({V'_{ans}}^{(i)}) = value({V'_{ans}}^{(pri)})$, then add vertex v_i to V''.

CLAIM 1: If for the ith SNDOP-query, if $value({V'_{ans}}^{(i)}) = value({V'_{ans}}^{(pri)})$, then v_i must be in the solution to SNDOP-ALL.
Suppose, by way of contradiction, that for the ith query, $value({V'_{ans}}^{(i)}) = value({V'_{ans}}^{(pri)})$, but v_i is not in the solution to SNDOP-ALL. Then, there is no V' of size $\leq k$ s.t. $v_i \in V'$ and V' is an answer to a the primary SNDOP-query. However, this is a contradiction, as given v_i and the vertices returned by the ith query, we are guaranteed this to be a valid answer to the primary query.
CLAIM 2: For each v_i in a solution to SNDOP-ALL, the ith SNDOP query returns a value s.t. $value({V'_{ans}}^{(i)}) = value({V'_{ans}}^{(pri)})$.
Suppose, by way of contradiction, that there is some v_i in the solution to SNDOP-ALL s.t. the ith query returns a value that is not equal to the value returned by the primary. However, by the definition of SNDOP-ALL, this is not possible, hence a contradiction.
PROOF OF PROPOSITION: Note the construction can be accomplished in PTIME. The proposition follows directly from claims 1-2. □

B. PROOFS FOR SECTION 5

B.1. Proof of Proposition 5.4

Suppose Π is any GAP. Then:

(1) $\mathbf{S}_\Pi$ is monotonic.
(2) $\mathbf{S}_\Pi$ has a least fixpoint $lfp(\mathbf{S}_\Pi)$ and $lfp(\mathbf{T}_\Pi) = grd(lfp(\mathbf{S}_\Pi))$. That is, $lfp(\mathbf{S}_\Pi)$ is a non-ground representation of the (ground) least fixpoint operator $\mathbf{T}_\Pi$.

PROOF. Part 1 follows directly from the definition – for a given atom A and interpretation I, $\mathbf{S}(I)(A) \geq I(A)$.
Part 2 follows directly from the definitions of $\mathbf{S}$ and $\mathbf{T}$. □

B.2. Proof of Theorem 5.6

Given SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$ and a GAP Π embedding a social network $\mathcal{S}$, if agg is monotonic then:

• There is an answer to the SNDOP query Q w.r.t. Π iff SNDOP-Mon$(\Pi, agg, VC, k, g_I(V), g_O(V))$ does not return NIL.
• If SNDOP-Mon$(\Pi, agg, VC, k, g_I(V), g_O(V))$ returns any result other than NIL, then that result is an answer to the SNDOP query Q w.r.t. Π.

PROOF. Part 1 ($\Leftarrow$): Suppose there is an answer to the query and SNDOP-Mon returns NIL. Then there is some set of vertices, sol of cardinality $\leq k$, s.t. $\Pi \cup \bigcup_{v \in sol} g_I(v) : 1 \models VC$. However, such a set would obviously have been added as a tuple into $Todo$ at step 2 or step 4(c)iB. Hence, a contradiction.
Part 1 ($\Rightarrow$): Suppose there is no answer to the query and SNDOP-Mon returns NIL. Then, there is no set of vertices, sol of cardinality $\leq k$, s.t. $\Pi \cup \bigcup_{v \in sol} g_I(v) : 1 \models VC$. SNDOP-Mon performs such a check at line 4b. Hence, a contradiction.

Part 2: Suppose, BWOC, there exists a set of vertices that is a solution, sol, of cardinality $\leq k$, s.t. $\bigcup_{v \in sol}\{g_I(v) : 1\}$ is not what is returned by SNDOP-Mon and $value(\Pi \cup \bigcup_{v \in sol}\{g_I(v) : 1\}$ is greater than $bestVal$. We note that SNDOP-Mon considers most sets of vertices of cardinality $\leq k$. Further, the monotonicity of agg and line 4(c)i tell us that the only solutions not considered are ones guaranteed to have a $value$ less than $bestVal$ – hence, a contradiction. □

B.3. Proof of Proposition 5.7

Given a SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$, a social network $\mathcal{S}$, and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$, the complexity of GREEDY-SNDOP is $O(k \cdot |\mathbf{V}| \cdot F(|\mathbf{V}|))$ where $F(|\mathbf{V}|)$ is the time complexity to compute $value(\mathbf{V}')$ for some set $\mathbf{V}' \subseteq \mathbf{V}$ of size k.

PROOF. The outer loop at line 2 iterates k times, the inner loop at line 2b iterates $O(|\mathbf{V}|)$ times, and at each inner loop, at line 2(b)i, the function $value$ is computed with costs $F(|\mathbf{V}|)$. The statement follows. □

B.4. Proof of Theorem 5.8

Given a SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$, a social network $\mathcal{S}$, and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$, if

— Π is a linear GAP
— Q is a-priori VC
— agg is positive-linear
— $value$ is zero-starting.

then GREEDY-SNDOP is an $(\frac{e}{e-1})$-approximation algorithm.

PROOF. [Nemhauser et al. 1978] proposes a greedy algorithm to solve the general problem of finding an element of a uniform matroid that maximizes a non-decreasing, submodular function F (defined over the elements of the matroid) s.t. $F(\emptyset) = 0$. The algorithm is very simple, yet guarantees an $\frac{e}{e-1}$ approximation: it incrementally builds a solution (without backtracking) starting with the empty set; in each iteration it adds an element that most improves the current solution (according to F).

Answering a SNDOP query is the problem of finding a pre-answer with maximum $value$. We show that, under the assumptions stated in the claim, the set of pre-answers is a uniform matroid and $value$ satisfies the restrictions stated above for F (which enables us to use the greedy algorithm of [Nemhauser et al. 1978]). The hypothesis that Q is a-priori VC entails that the set of pre-answers is a uniform matroid by Lemma 3.14. The hypothesis agg is positive-linear entails that agg is monotonic (see Proposition 3.9); the latter in turn entails that $value$ is monotonic (see Lemma 3.13). The first, second, and third hypotheses in the claim entail that $value$ is submodular, by Theorem 3.15. Recall that the fourth hypothesis means $value(\emptyset) = 0$ by definition.

Hence, under the conditions stated in the claim, answering a SNDOP query is an instance of the problem addressed in [Nemhauser et al. 1978]. Since GREEDY-SNDOP is a specialization of the algorithm in [Nemhauser et al. 1978] applied to our setting, it follows that GREEDY-SNDOP is an $(\frac{e}{e-1})$-approximation algorithm. □